# Tóm tắt đồ án

**XÂY DỰNG HỆ HỖ TRỢ TƯ VẤN ĐỊA ĐIỂM DU DỊCH TẠI HUẾ**

**TÓM TẮT:** Ngày nay, với sự phát triển mạnh mẽ của khoa học máy tính, việc ứng dụng học máy để xây hệ thống tư vấn cho thấy những ưu điểm nổi trội so với các hình thức tư vấn truyền thống. Tính chất động và phức tạp của sở thích du lịch được khám phá thông qua việc phân tích năm đặc trưng chính: mức thu nhập, địa hình, thời gian, mục đích du lịch, và số lượng người đi cùng. Tập dữ liệu đa dạng và chi tiết được thu thập từ khảo sát người dùng, bao gồm cả thông tin về đặc trưng cá nhân và đánh giá của họ về các địa điểm du lịch tại Huế. Bằng cách này, ứng dụng học máy có thể dựa vào thông tin đặc trưng của người dùng và đề xuất những địa điểm phù hợp nhất. Bài báo cáo mô tả chi tiết quá trình xây dựng mô hình học máy, từ việc lựa chọn tập dữ liệu đến thử nghiệm với các mô hình khác nhau như SVM, Xgboost, Random Forest, và Decision Tree. Kết quả thử nghiệm chỉ ra rằng SVM là mô hình hiệu quả nhất cho ứng dụng này. Mô hình đưa ra dự đoán về đánh giá của người dùng đối với từng địa điểm, từ đó ứng dụng có thể gợi ý 3 địa điểm có đánh giá cao nhất. Bằng cách này, ứng dụng web không chỉ cung cấp trải nghiệm du lịch cá nhân hóa mà còn thúc đẩy khám phá địa điểm mới dựa trên ước lượng chính xác về sở thích và tiêu chí du lịch của người dùng.

*Từ khóa: Tư vấn địa điểm du lịch, Học máy, Phân lớp*

**ABSTRACT**: Today, with the strong development of computer science, the application of machine learning to build consulting systems shows outstanding advantages compared to traditional forms of consulting. The dynamic and complex nature of travel preferences is explored through the analysis of five main characteristics: income level, terrain, time, travel purpose, and number of accompanying people. A diverse and detailed data set is collected from user surveys, including information about personal characteristics and their assessments of tourist destinations in Hue. In this way, machine learning applications can rely on user characteristic information and recommend the most suitable locations. The report describes in detail the process of building a machine learning model, from selecting the data set to testing with different models such as SVM, Xgboost, Random Forest, and Decision Tree. Experimental results indicate that SVM is the most effective model for this application. The model makes predictions about user ratings for each location, from which the application can suggest 3 locations with the highest ratings. In this way, the web application not only provides a personalized travel experience but also promotes new place discovery based on an accurate estimate of the user's preferences and travel criteria.

***Keywords:*** *Tourist destination consulting, Machine learning, Classification*

# Danh mục các kí hiệu, chữ cái viết tắt

|  |  |  |
| --- | --- | --- |
| **STT** | **Kí hiệu/Viết tắt** | **Diễn giải** |
| 1 | DSS | Decision Support System |
| 2 | SVM | Support Vector Machines |
| 3 | JS | Javascript |
| 4 | CSDL | Cơ sở dữ liệu |
| 5 | ML | Machine learning |
| 6 | TP | True Positive |
| 7 | TN | True Negative |
| 8 | FP | False Positive |
| 9 | FN | False Negative |
| 10 | SMOTE | Synthetic Minority Over-sampling Technique |

# Danh mục bảng biểu

[Bảng 2. 1: Bảng kết quả hiện chỉnh tham số mô hình Cây quyết định 55](#_Toc153490362)

[Bảng 2. 2: Bảng kết quả hiện chỉnh tham số mô hình Rừng cây ngẫu nhiên 61](#_Toc153490363)

[Bảng 2. 3: Bảng kết quả hiện chỉnh tham số mô hình XGBoost 64](#_Toc153490364)

[Bảng 2. 4: Kết quả đánh giá mô hình Decision Tree trên tập dữ liệu kiểm thử 68](#_Toc153490365)

[Bảng 2. 5: Kết quả đánh giá mô hình XBGoost trên tập dữ liệu kiểm thử 74](#_Toc153490366)

[Bảng 2. 6: Kết quả đánh giá mô hình XBGoost trên tập dữ liệu kiểm thử 76](#_Toc153490367)

[Bảng 3. 1: Kết quả hiện chỉnh tham số các mô hình học máy 88](#_Toc153490480)

[Bảng 3. 2: Kết quả đánh giá các mô hình học máy 89](#_Toc153490481)

# Danh mục hình ảnh

[Hình 1. 1: Cấu trúc hệ hỗ trợ ra quyết định 7](#_Toc153490368)

[Hình 1. 2: Phân loại Machine learning 11](#_Toc153490369)

[Hình 1. 3: Margin trong SVM 12](#_Toc153490370)

[Hình 1. 4: Siêu mặt phẳng trong thuật toán phân lớp SVM 14](#_Toc153490371)

[Hình 1. 5: Cấu trúc cây quyết định 15](#_Toc153490372)

[Hình 1. 6: Ví dụ về bài toán phân lớp sử dụng decision tree. 16](#_Toc153490373)

[Hình 1. 7: Đồ thị của hàm entropy với n=2 17](#_Toc153490374)

[Hình 1. 8: Mô tả thuật toán Bagging và Boosting 19](#_Toc153490375)

[Hình 1. 9: Mô tả thuật toán Rừng cây ngẫu nhiên 21](#_Toc153490376)

[Hình 1. 10: Mô tả thuật toán Gradient Boosting 23](#_Toc153490377)

[Hình 1. 11: Mô hình thuật toán Gradient Tree Boosting 23](#_Toc153490378)

[Hình 1. 12: Ưu điểm của thuật toán Xgboost 27](#_Toc153490379)

[Hình 1. 13: Mô tả confusion matrix 30](#_Toc153490380)

[Hình 1. 14: Mô tả chỉ số Precision và Recall 32](#_Toc153490381)

[Hình 1. 15: Luồng thực hiện dự án học máy 35](#_Toc153490382)

[Hình 2. 1: Cài đặt Anaconda bước 1 38](#_Toc153490383)

[Hình 2. 2: Cài đặt Anaconda bước 2 39](#_Toc153490384)

[Hình 2. 3: Cài đặt Anaconda bước 3 39](#_Toc153490385)

[Hình 2. 4: Cài đặt Anaconda bước 4 40](#_Toc153490386)

[Hình 2. 5: Cài đặt Anaconda bước 5 40](#_Toc153490387)

[Hình 2. 6: Cài đặt Anaconda bước 6 41](#_Toc153490388)

[Hình 2. 7: Cài đặt Anaconda bước 7 41](#_Toc153490389)

[Hình 2. 8: Cài đặt Anaconda bước 8 42](#_Toc153490390)

[Hình 2. 9: Cài đặt Python 43](#_Toc153490391)

[Hình 2. 10: Logo thư viện Scikit-learn 44](#_Toc153490392)

[Hình 2. 11: Thư viện Flask 46](#_Toc153490393)

[Hình 2. 12: Tệp requirements.txt chứa thông tin các thư viện 47](#_Toc153490394)

[Hình 2. 13: Trực quan hóa dữ liệu về Mức đánh giá 50](#_Toc153490395)

[Hình 2. 14: Phân chia tập dữ liệu 51](#_Toc153490396)

[Hình 2. 15: Tập dữ liệu huấn luyện theo Mức đánh giá 52](#_Toc153490397)

[Hình 2. 16: Dữ liệu sau khi mã hóa One-hot 53](#_Toc153490398)

[Hình 2. 17: Trực quan dữ liệu theo Mức đánh giá sau khi cân bằng 54](#_Toc153490399)

[Hình 2. 18: Trực quan hóa dữ liệu test theo Mức đánh giá 67](#_Toc153490400)

[Hình 2. 19: Mã trận lỗi của mô hình Decision Tree 70](#_Toc153490401)

[Hình 2. 20: Ma trận lỗi mô hình Rừng cây quyết định 72](#_Toc153490402)

[Hình 2. 21: Ma trận lỗi mô hình Xgboost trên tập dữ liệu kiểm thử 75](#_Toc153490403)

[Hình 2. 22: Ma trận lỗi mô hình SVM đối với tập dữ liệu kiểm thử 77](#_Toc153490404)

[Hình 2. 23: Cấu trúc dự án Flask 78](#_Toc153490405)

[Hình 3. 1: Kết quả tập dữ liệu thu thập được 85](#_Toc153490482)

[Hình 3. 2: Kết quả tập dữ liệu sau khi tiền xử lí 86](#_Toc153490483)

[Hình 3. 3: Kết quả tập dữ liệu sau khi mã hóa 87](#_Toc153490484)

[Hình 3. 4: Kết quả tập dữ liệu sau khi cân bằng 88](#_Toc153490485)

[Hình 3. 5: Kết quả giao diện nhập thông tin người dùng 89](#_Toc153490486)

[Hình 3. 6: Kết quả giao diện hiển thị kết quả gợi ý 89](#_Toc153490487)

[Hình 3. 7: Kết quả giao diện ứng dụng web 90](#_Toc153490488)

[Hình 3. 8: Kết quả giao diện tìm kiếm khi ấn vào kết quả 91](#_Toc153490489)

# Mục lục

[Tóm tắt đồ án](#_Toc153490635)

[Danh mục các kí hiệu, chữ cái viết tắt](#_Toc153490636)

[Danh mục bảng biểu](#_Toc153490637)

[Danh mục hình ảnh](#_Toc153490638)

[Mục lục](#_Toc153490639)

[Phần mở đầu 1](#_Toc153490640)

[1. Lí do chọn đề tài 1](#_Toc153490641)

[2. Mục tiêu của đề tài 1](#_Toc153490642)

[3. Đối tượng và phạm vi nghiên cứu 2](#_Toc153490643)

[4. Phương pháp nghiên cứu 2](#_Toc153490644)

[5. Ý nghĩa khoa học và thực tiến của đề tài 3](#_Toc153490645)

[6. Cấu trúc báo cáo 3](#_Toc153490646)

[Chương 1: Tổng quan về hệ hỗ trợ tư vấn địa điểm du lịch 5](#_Toc153490647)

[1.1. Hệ hỗ trợ ra quyết định 5](#_Toc153490648)

[1.1.1. Khái niệm 5](#_Toc153490649)

[1.1.2. Đặc trưng 5](#_Toc153490650)

[1.1.3. Cấu trúc 5](#_Toc153490651)

[1.2. Học máy 7](#_Toc153490652)

[1.2.1. Khái niệm 7](#_Toc153490653)

[1.2.2. Ứng dụng 7](#_Toc153490654)

[1.2.3. Phân loại 9](#_Toc153490655)

[1.3. Bài toán dự đoán ứng dụng mô hình học máy 11](#_Toc153490656)

[1.3.1. Thuật toán Máy vector hỗ trợ 12](#_Toc153490657)

[1.3.2. Thuật toán Cây quyết định 14](#_Toc153490658)

[1.3.3. Thuật toán Học kết hợp 18](#_Toc153490659)

[1.4. Đánh giá mô hình phân lớp 27](#_Toc153490660)

[1.4.1. Các phương pháp đánh giá mô hình học máy 28](#_Toc153490661)

[1.4.2. Đánh giá mô hình phân lớp 29](#_Toc153490662)

[1.4.3. Đánh giá mô hình học máy với dữ liệu mất cân bằng 31](#_Toc153490663)

[1.5. Quy trình triển khai một dự án học máy 34](#_Toc153490664)

[Chương 2: Xây dựng ứng dụng hỗ trợ gợi ý địa điểm du lịch tại Huế 38](#_Toc153490665)

[2.1. Môi trường và công cụ 38](#_Toc153490666)

[2.1.1. Cài đặt Anaconda 38](#_Toc153490667)

[2.1.2. Cài đặt Python 42](#_Toc153490668)

[2.1.3. Cài đặt Scikit-learn 44](#_Toc153490669)

[2.1.4. Cài đặt XGBoost 45](#_Toc153490670)

[2.1.5. Cài đặt Flask 45](#_Toc153490671)

[2.3. Tạo tập dữ liệu 47](#_Toc153490672)

[2.3.1. Thu thập dữ liệu 47](#_Toc153490673)

[2.3.2. Xử lí dữ liệu 48](#_Toc153490674)

[2.3.3. Phân chia tập dữ liệu 50](#_Toc153490675)

[2.3.4. Mã hóa dữ liệu 52](#_Toc153490676)

[2.3.5. Cân bằng dữ liệu 53](#_Toc153490677)

[2.4. Hiệu chỉnh tham số mô hình học máy 54](#_Toc153490678)

[2.4.1. Chọn tham số cho mô hình Cây quyết định 54](#_Toc153490679)

[2.4.2. Chọn tham số cho mô hình Rừng cây ngẫu nhiên 60](#_Toc153490680)

[2.4.3. Chọn tham số cho mô hình Xgboost 63](#_Toc153490681)

[2.4.4. Chọn tham số cho mô hình Máy vector hỗ trợ 66](#_Toc153490682)

[2.5. Huấn luyện và đánh giá mô hình 67](#_Toc153490683)

[2.5.1. Đánh giá mô hình Cây quyết định 67](#_Toc153490684)

[2.5.2. Đánh giá mô hình Rừng cây ngẫu nhiên 70](#_Toc153490685)

[2.5.3.. Đánh giá mô hình XGBoost 72](#_Toc153490686)

[2.5.4. Đánh giá mô hình Máy hỗ trợ vector 75](#_Toc153490687)

[2.6. Xây dựng giao diện ứng dụng 78](#_Toc153490688)

[2.6.1. Cấu trúc chương trình 78](#_Toc153490689)

[2.6.2. Phía Fontend 78](#_Toc153490690)

[2.6.3. Phía Backend 82](#_Toc153490691)

[2.6.4. Tương tác giữa Frontend và Backend 83](#_Toc153490692)

[Chương 3: Kết quả đạt được 85](#_Toc153490693)

[3.1. Kết quả thu thập và xử lí dữ liệu 85](#_Toc153490694)

[3.2. Kết quả hiệu chỉnh tham số các mô hình học máy 88](#_Toc153490695)

[3.2. Kết quả huấn luyện và đánh giá các mô hình học máy 89](#_Toc153490696)

[3.3. Kết quả xây dựng ứng dụng 89](#_Toc153490697)

[Chương 4: Kết luận và hướng phát triển 92](#_Toc153490698)

[4.1. Kết quả đạt được 92](#_Toc153490699)

[4.2. Hạn chế 92](#_Toc153490700)

[4.3. Hướng phát triển 92](#_Toc153490701)

[Tài liệu tham khảo 94](#_Toc153490702)

[Phụ lục 95](#_Toc153490703)

[Phụ lục 1: Mã nguồn HTML 95](#_Toc153490704)

[Phụ lục 2: Mã nguồn Javascript 96](#_Toc153490705)

[Phụ lục 3: Mã nguồn CSS 99](#_Toc153490706)

[Phụ lục 4: Mã nguồn chương trình chính 101](#_Toc153490707)

[Phụ lục 5: Mã nguồn huấn luyện mô hình học máy 103](#_Toc153490708)

# Phần mở đầu

## Lí do chọn đề tài

Du lịch đang trở thành một phần quan trọng của cuộc sống hiện đại, và người du lịch ngày càng mong muốn có những trải nghiệm độc đáo và phù hợp với sở thích cá nhân của họ. Tuy nhiên, việc chọn địa điểm du lịch phù hợp với mong muốn và nhu cầu cá nhân có thể là một thách thức đối với du khách, đặc biệt là khi họ đến một thành phố lớn với nhiều điểm du lịch khác nhau.

Qua khảo sát thực tế, nhận thấy nhu cầu của du khách không chỉ dừng lại ở việc khám phá những điểm du lịch nổi tiếng, mà còn muốn trải nghiệm những địa điểm độc đáo và phù hợp với cá nhân họ. Cần một hệ thống hỗ trợ quyết định thông minh, có khả năng tư vấn và đề xuất những điểm đến tương xứng với sở thích và mong muốn của từng du khách.

Huế, với vẻ đẹp lịch sử và văn hóa đặc sắc, là một trong những điểm đến du lịch hàng đầu tại Việt Nam. Để giúp du khách trải nghiệm tối đa và lựa chọn được những địa điểm phù hợp với sở thích cá nhân, việc xây dựng một hệ thống hỗ trợ ra quyết định sử dụng kỹ thuật học máy trở nên quan trọng. Nó mang lại một số lợi ích thiết thực như:

- Tối ưu hóa trải nghiệm du lịch: Hệ thống sẽ giúp du khách tận hưởng những điểm du lịch phù hợp với sở thích, tạo ra trải nghiệm du lịch cá nhân hóa và độc đáo.

- Tiết kiệm thời gian và nỗ lực: Dựa trên dữ liệu và kỹ thuật học máy, hệ thống sẽ cung cấp gợi ý nhanh chóng và hiệu quả, giúp du khách tiết kiệm thời gian trong việc lựa chọn địa điểm.

- Phát triển du lịch bền vững: Hỗ trợ du khách lựa chọn các địa điểm du lịch hướng đến phát triển bền vững, góp phần vào việc bảo vệ và giữ gìn di sản văn hóa và tự nhiên.

Lựa chọn đề tài "Xây dựng hệ hỗ trợ tư vấn địa điểm du lịch tại huế sử dụng kỹ thuật học máy" xuất phát từ nhận thức sâu sắc về nhu cầu ngày càng cao của du khách trong việc trải nghiệm du lịch một cách cá nhân hóa và bền vững.

## 2. Mục tiêu của đề tài

Mục tiêu chính của đề tài "Xây dựng hệ hỗ trợ tư vấn địa điểm du lịch tại huế sử dụng kỹ thuật học máy" là phát triển và triển khai một hệ thống thông minh nhằm đưa ra gợi ý về địa điểm du lịch phù hợp từ thông tin cá nhân của người dùng, qua đó tối ưu hóa trải nghiệm du lịch cho du khách tại thành phố Huế. Dưới đây là những mục tiêu cụ thể của đề tài:

- Thu thập, xây dựng bộ dữ liệu huấn luyện cho mô hình học máy

- Xử lí dư liệu phù hợp với đầu vào mô hình huấn luyện

- Nghiên cứu, đánh giá, lựa chọn mô hình học máy phù hợp với bộ dữ liệu: Support Vector Machine, Decision Tree, Random Forest, Xgboost.

- Xây dựng giao diện ứng dụng web

- Kiểm nghiệm ứng dụng, triển khai thực tế.

## 3. Đối tượng và phạm vi nghiên cứu

\* Đối tượng nghiên cứu

- Tập dữ liệu chứa thông tin cá nhân và đánh giá về 20 địa điểm du lịch tại Huế.

- Các thuật toán học máy phân loại, thuật toán tối ưu nhất được chọn. Ngôn ngữ lập trình Python, các thư viện hỗ trợ.

- Ứng dụng web, ngôn ngữ lập trình Javascript, HTML, CSS, các côn

\* Phạm vi nghiên cứu

- Phạm vi thu thập dữ liệu cá nhân chứa các thông tin cơ bản:

+ Họ tên: Thông tin Họ và tên của người được khảo sát

+ Thông tin nghề nghiệp, mức thu nhập

+ Thông tin về địa hình du lịch mong muốn tới

+ Thông tin về số lượng người tham gia du lịch

+ Thông tin về thời gian

+ Thông tin về mục đích du lịch

+ Địa điểm du lịch và đánh giá về địa điểm du lịch đó

- Phạm vi ứng dụng: Các địa điểm du lịch tại Huế

## 4. Phương pháp nghiên cứu

*\* Phương pháp lý thuyết*

- Tiến hành thu thập và nghiên cứu các tài liệu có liên quan đến công việc tư vấn hướng nghiệp.

- Nghiên cứu và vận dụng các kỹ thuật về khai phá dữ liệu, các thuật toán được áp dụng cho từng kỹ thuật.

- Nghiên cứu các thuật toán học máy cho bài toán phân lớp: SVM. Decision Tree, XGBoost.

*\* Phương pháp thực nghiệm*

- Xây dựng ứng dụng trên nền tảng web.

- Sử dụng back-end là ngôn ngữ Python với framework là Flask.

- Sử dụng front-end là HTML, CSS, Javascript.

- Các thuật toán học máy Support Vector Machine, Decision Tree, Random Forest, Xgboost sử dụng Python với thư viện Scikit-learn.

-Kiểm tra, thử nghiệm và đánh giá kết quả

## 5. Ý nghĩa khoa học và thực tiến của đề tài

\* Về mặt khoa học

- Hiểu được một số kỹ thuật cơ bản để khai phá dữ liệu, các chức năng và ứng dụng của khai phá dữ liệu.

- Phát triển ứng dụng để góp phần phổ biến và từng bước thâm nhập sâu hơn về kỹ thuật này.

\* Về mặt thực tiễn

- Ứng dụng những thành quả của công nghệ thông tin vào lĩnh vực du lịch, góp phần tối ưu hóa trải nghiệm du lịch cho du khách tại thành phố Huế. Dưới đây là những mục tiêu cụ thể của đề tài:

+ Tối ưu hóa trải nghiệm du lịch: Hệ thống sẽ giúp du khách tận hưởng những điểm du lịch phù hợp với sở thích, tạo ra trải nghiệm du lịch cá nhân hóa và độc đáo.

+ Tiết kiệm thời gian và nỗ lực: Dựa trên dữ liệu và kỹ thuật học máy, hệ thống sẽ cung cấp gợi ý nhanh chóng và hiệu quả, giúp du khách tiết kiệm thời gian trong việc lựa chọn địa điểm.

+ Phát triển du lịch bền vững: Hỗ trợ du khách lựa chọn các địa điểm du lịch hướng đến phát triển bền vững, góp phần vào việc bảo vệ và giữ gìn di sản văn hóa và tự nhiên.

## 6. Cấu trúc báo cáo

Ngoài phần mở đầu, danh mục hình ảnh, bảng biểu, kí hiệu viết tắt và danh mục tài liệu tham khảo, nội dung luận văn được trình bày thành 4 chương, cụ thể như sau:

- CHƯƠNG 1: TỔNG QUAN VỀ HỆ TƯ VẤN ĐỊA ĐIỂM DU LỊCH

Chương này trình bài cơ sở lí luận, thực tiễn của vấn đề, các công trình nghiên cứu có liên quan, các nội dung kiến thức có trong đề tài được luận giải rõ ràng, rành mạch, cụ thể.

- CHƯƠNG 2: XÂY DỰNG ỨNG DỤNG GỢI Ý ĐỊA ĐIỂM DU LỊCH TẠI HUẾ

Chương này trình bày nội dung chủ yếu của đề tài:xác định nội dung, cài đặt môi trường, công cụ, quá trình thu thập, xử lí dữ liệu, các bước thực hiện xây dựng hệ thống.

- CHƯƠNG 3: KẾT QUẢ ĐẠT ĐƯỢC

Chương này trình bày kết quả đạt được sau quá trình thực nghiệm, kết quả thu thập dữ liệu, sử dựng thuật toán, giao diện chương trình được xây dựng.

- CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

# Chương 1: Tổng quan về hệ hỗ trợ tư vấn địa điểm du lịch

## 1.1. Hệ hỗ trợ ra quyết định

### 1.1.1. Khái niệm

Trong thập niên 1970, Scott Morton đưa ra khái niệm đầu tiên về Hệ hỗ trợ ra quyết định (Decision Support System - DSS). Ông định nghĩa DSS như là những hệ thống máy tính tương tác nhằm giúp những người ra quyết định sử dụng dữ liệu và mô hình để giải quyết các vấn đề không có cấu trúc hoặc bán cấu trúc.

### 1.1.2. Đặc trưng

Thông thường các hệ hỗ trợ quyết định có đặc trưng gồm:

- Phần mềm máy tính

- Chức năng hỗ trợ ra quyết định

- Làm việc với các bài toán có cấu trúc yếu

- Hoạt động theo cách tương tác với người dùng

- Được trang bị nhiều mô hình phân tích và mô hình dữ liệu.

Ưu thế của người ra quyết định: Kinh nghiệm; Khả năng trực giác; Có óc phán đoán; Có tri thức tổng hợp

Ưu thế của máy tính: Tốc độ; Thông tin; Khả năng xử lý

Kết hợp cả ưu thế của người ra quyết định và máy tính, ta có ưu thế của Hệ hỗ trợ ra quyết định:

- Tăng hiệu quả

- Tăng sự hiểu biết

- Tăng tốc độ

- Tăng tính linh hoạt

- Giảm sự phức tạp

- Giảm chi phí

Hiện tại chưa có một định nghĩa thống nhất nào về DSS. Tuy nhiên tất cả đều đồng ý mục đích cơ bản nhất của DSS là để hỗ trợ và cải tiến việc ra quyết định.

### 1.1.3. Cấu trúc

Cấu trúc một hệ hỗ trợ ra quyết định gồm có bốn thành phần chính sau:

- Nhận lực

- Phần mềm

- Mô hình

- Cơ sở dữ liệu

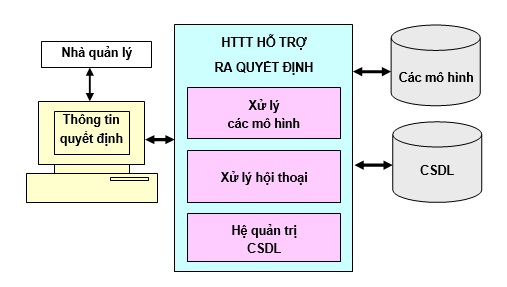
Chức năng cụ thể của từng thành phần này như sau:

\* Nhân lực: Bao gồm nhà quản lí sử dụng hệ thống, các lập trình viên và nhân viên kí thuật quản lí hệ thống

\* Phần mềm: Bao gồm các module để quản lý CSDL, các mô hình thông qua quyết định và các chế độ hội thoại giữa người sử dụng với hệ thống. Hệ thống phần mềm DSS cho phép người sử dụng có thể can thiệp vào CSDL và cơ sở mô hình của hệ thống một cách dễ dàng. Hệ thống phần mềm DSS cung cấp các biểu đồ dễ sử dụng và các giao diện linh hoạt, cho phép cả các nhà quản lý không có kinh nghiệm sử dụng máy tính cũng có thể tiếp cận hệ thống một cách dễ dàng

\* Mô hình: tổng thể các mô hình phân tích và toán học sử dụng trong quá trình ra quyết định như mô hình thống kê, mô hình dự báo, mô hình điều hành, mô hình lập kế hoạch. Mỗi mô hình là một sự mô tả các yếu tố hoặc các mối quan hệ của một hiện tượng nào đó, ví dụ như mô hình phân tích hồi quy, phân tích độ nhạy, phân tích tình huống, tìm điểm hoà vốn, mô hình bài toán quy hoạch tuyến tính… Mỗi hệ thống DSS được xây dựng cho một tập hợp các mục đích khác nhau và sẽ tạo ra một tập hợp các mô hình phụ thuộc theo mục đích mà nó hướng tới.

\* Cơ sở dữ liệu(CSDL): bao gồm dữ liệu (hiện tại hoặc quá khứ) từ CSDL của các tổ chức kinh tế, ngân hàng dữ liệu bên ngoài, CSDL nội bộ riêng cho các nhà quản lý. Hệ thống DSS bảo đảm tính toàn vẹn của dữ liệu, bản thân nó không tạo ra hoặc cập nhật dữ liệu mà chỉ tổ chức dữ liệu lại theo cách mà từng cá nhân hoặc từng nhóm nhận thấy là phù hợp để tạo quyết định dựa trên tình trạng thực tế. CSDL của mô hình này thường đã được tổng hợp và lưu trữ đặc biệt cho mục đích sử dụng riêng của hệ thống DSS do hai nguyên nhân: tổ chức cần bảo vệ CSDL của tổ chức, chống sự phá hoại từ những thay đổi đột ngột hoặc không thích hợp; nếu tự rà soát CSDL lớn của tổ chức thì đó sẽ là một quá trình chậm chạp và tốn kém.



Hình 1. : Cấu trúc hệ hỗ trợ ra quyết định

## 1.2. Học máy

### 1.2.1. Khái niệm

Học máy - Machine learning (ML) là một lĩnh vực con của Trí tuệ nhân tạo (Artificial Intelligence) sử dụng các thuật toán cho phép máy tính có thể học từ dữ liệu để thực hiện các công việc thay vì được lập trình một cách rõ ràng.

Trong cuốn sách của mình [2], Tom Mitchell đã định nghĩa machine learning: “*A computer program is said to learn to perform a****task T****from****experience E****, if its performance at task T, as measured by a****performance metric P****, improves with experience E over time”*

Tạm dịch: “*Một chương trình máy tính được cho là học để thực hiện một****nhiệm vụ T****từ****kinh nghiệm E****, nếu hiệu suất thực hiện công việc T của nó được đo bởi****chỉ số hiệu suất P****và được cải thiện bởi kinh nghiệm E theo thời gian*”

Định nghĩa trên được dùng phổ biến và cho ta hình dung một các rõ ràng các phần của một thuật toán ML. Nó gồm 3 yếu tố cơ bản: nhiệm vụ T, kinh nghiệm E, chỉ số hiệu suất P. Dựa vào các yếu tố này mà người ta phân loại có ML theo nhiều các khác nhau, ứng dụng nhiều lĩnh vực trên thực tế

### 1.2.2. Ứng dụng

Trên thực tế, hiện nay Machine learning đang là lĩnh vực cho thấy sự tiến bộ trong tiến trình của Trí tuệ nhân tạo. Nó đang là một chủ đề nóng và có khả năng làm cho máy móc trở nên thông minh hơn. Một số lĩnh vực phổ biến mà ML góp mặt như:

*\* Xử lí ảnh*

Bài toán xử lý ảnh (Image Processing) giải quyết các vấn đề phân tích thông tin từ hình ảnh hay thực hiện một số phép biến đổi. Một số ví dụ là:

**- Gắn thẻ hình ảnh** (Image Tagging), giống như Facebook, một thuật toán tự động phát hiện khuôn mặt của bạn và bạn bè trên những bức ảnh. Về cơ bản, thuật toán này học từ những bức ảnh mà bạn tự gắn thẻ cho mình trước đó.

**- Nhận dạng ký tự** (Optical Character Recognition), là một thuật toán chuyển dữ liệu trên giấy tờ, văn bản thành dữ liệu số hóa. Thuật toán phải học cách nhận biết ảnh chụp của một ký tự là ký tự nào.

**- Ô tô tự lái** (Self-driving cars), một phần cơ chế sử dụng ở đây là xử lý ảnh. Một thuật toán machine learning giúp phát hiện các mép đường, biển báo hay các chướng ngại vật bằng cách xem xét từng khung hình video từ camera.

*\* Phân tích văn bản*

Phân tích văn bản (Text analysis) là công việc trích xuất hoặc phân lọi thông tin từ văn bản. Các văn bản ở đây có thể là các facebook posts, emails, các đoạn chats, tài liệu, … Một số ví dụ phổ biến là:

**- Lọc spam** (Spam filtering), là một trong những ứng dụng phân loại văn bản được biết và sử dụng nhiều nhất. Ở đây, phân loại văn bản là xác định chủ đề cho một văn bản. Bộ lọc spam sẽ học cách phân loại một email có phải spam không dựa trên nội dung và tiêu đề của email.

**- Phân tích ngữ nghĩa** (Sentiment Analysis), học cách phân loại một ý kiến là tích cực, trung tính hay tiêu cực dựa trên nội dung văn bản của người viết.

**- Khai thác thông tin** (Information Extraction), từ một văn bản, học cách để trích xuất các thông tin hữu ích. Chẳng hạn như trích xuất địa chỉ, tên người, từ khóa, …

*\* Khai phá dữ liệu*

Khai phá dữ liệu (Data mining) là quá trình khám phá ra các thông tin có giá trị hoặc đưa ra các dự đoán từ dữ liệu. Định nghĩa này có vẻ bao quát, nhưng bạn hãy nghĩ về việc tìm kiếm thông tin hữu ích từ một bảng dữ liệu rất lớn. Mỗi bản ghi sẽ là một đối tượng cần phải học, và mỗi cột là một đặc trưng. Chúng ta có thể dự đoán giá trị của một cột của bản ghi mới dựa trên các bản ghi đã học. Hoặc là phân nhóm các bản ghi của bản. Sau đây là những ứng dụng của khai phá dữ liệu:

**- Phát hiện bất thường** (Anomaly detection), phát hiện các ngoại lệ, ví dụ như phát hiện gian lận thẻ tín dụng. Bạn có thể phát hiện một giao dịch là khả nghi dựa trên các giao dịch thông thường của người dùng đó.

**- Phát hiện các quy luật** (Association rules), ví dụ, trong một siêu thị hay một trang thương mại điện tử. Bạn có thể khám phá ra khách hàng thường mua các món hàng nào cùng nhau. Dễ hiểu hơn, khách hàng của bạn khi mua món hàng A thường mua kèm món hàng nào? Các thông tin này rất hữu ích cho việc tiếp thị sản phẩm.

**- Gom nhóm** (Grouping), ví dụ, trong các nền tảng SaaS, người dùng được phân nhóm theo hành vi hoặc thông tin hồ sơ của họ.

**- Dự đoán** (Predictions), các cột giá trị (của một bản ghi mới trong database). Ví dụ, bạn có thể dự đoán giá của căn hộ dựa trên các dữ liệu về giá các căn hộ bạn đã có.

*\* Trò chơi điện tử & Robot*

*T*rò chơi điện tử (Video games) và robot (Robotics) là lĩnh vực lớn có sự góp mặt của machine learning. Nếu ta có một nhân vật cần di chuyển và tránh các chướng ngại vật trong game. Machine learning có thể học và giải quyết công việc này thay bạn. Một kỹ thuật phổ biến được áp dụng trong trường hợp này là Học tăng cường (Reinforcement learning). Ở đó, máy sẽ học tăng cường với mục tiêu là giải quyết nhiệm vụ trên. Học tăng cường là tiêu cực nếu nó va phải chướng ngại vật, là tích cực nếu nó chạm tới đích.

Một thành tựu gần đây nhất là cỗ máy Alpha Go của Google DeepMind đã đánh bại kỳ thủ cờ vậy số 1 thế giới. Trong khi cờ vây là một trò chơi có không gian trạng thái cực kỳ lớn.

### 1.2.3. Phân loại

Học máy có thể được phân làm 3 loại: Học có dám sát (Supervied Learning), Học không dám sát (Unsupervised Learning), Học tăng cường (Reinforcement Learning). Ngoài khác nhau về cách thuật toán sử dụng dữ liệu và loại vấn đề mà chúng giải quyết thì khác biệt lớn giữa các loại thuật toán này là cách mà chúng ta cung cấp tập dữ liệu huấn luyện cho mô hình.

*\* Học có giám sát:*

Trong học có giám sát, thuật toán machine learning thực hiện quá trình **“chuyển dữ liệu đầu vào thành đầu ra mong muốn”.**

Đầu vào được gọi là input, đầu ra còn được gọi là  nhãn (label). Tập dữ liệu huấn luyện được gọi là training set; Mỗi mẫu dữ liệu trong tập huấn luyện được gọi là training example.

Thuật toán học có giám sát cần học cách để chuyển đổi mỗi input (đầu vào) thành nhãn (kết quả) tương ứng. Do vậy, mọi mẫu dữ liệu huấn luyện trong tập dữ liệu huấn luyện cần phải biết trước nhãn của nó.

Tùy thuộc vào loại đầu ra mong muốn, chúng ta tiếp tục chia nhỏ học có giám sát gồm:

- Phân loại (Classification): Khi đầu ra mong muốn của chúng ta là một tập hữu hạn và rời rạc. Chẳng hạn như là giá trị cảm xúc của khách hàng: tích cực, tiêu cực, trung tính.

- Hồi quy (Regression):Khi đầu ra mong muốn là một dải giá trị liên tục. Về mặt bản chất hồi quy là bài toán gán nhãn cho dữ liệu thực, biểu diễn và dự đoán đầu ra dựa trên tổng quát hóa các dữ liệu từ đầu vào để tìm ra một hàm dự đoán. Chẳng hạn xây dựng hàm dự đoán giá nhà, giá cổ phiếu theo thời gian hoặc các biến đầu vào khác.

Nếu ta chú ý có thể thấy rằng, thực chất bài toán phân lớp cũng là một trường hợp đặc biệt của bài toán hồi quy, khi giá trị các dự đoán đầu ra thay vì các lớp thì là các giá trị rời rạc đại diện cho lớp.

Học có giám sát là thuật toán phổ biến nhất trong các thuật toán machine learning. Hạn chế khi sử dụng thuật toán này là chúng ta cần cung cấp dữ liệu có gán nhãn. Trong nhiều trường hợp, để có được dữ liệu gán nhãn này rất tốn rất nhiều chi phí. Chẳng hạn, nếu ta cần 10.000 review có nhãn để huấn luyện mô hình; Việc này sẽ cần con người đọc từng review và gán nhãn thủ công; Điều này rất tốn thời gian và công sức. Đây cũng là một rào cản của ML: xây dựng các tập dữ liệu gán nhãn chất lượng.

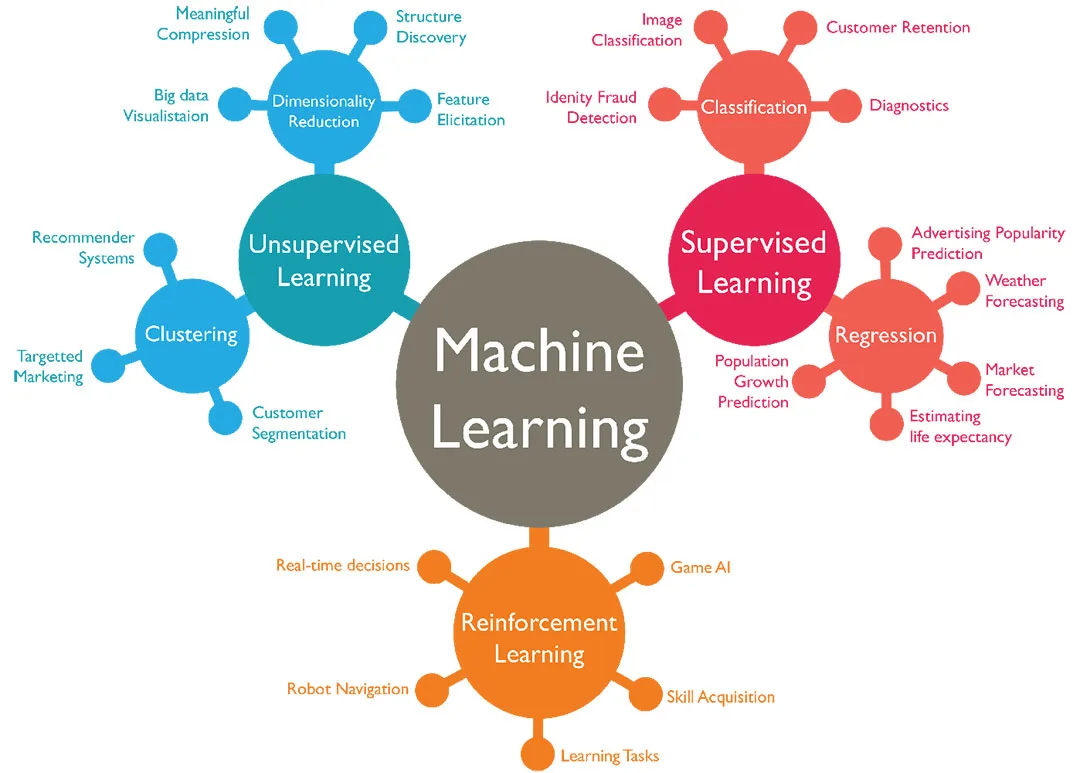
*\*Học không giám sát*

Trong học không giám sát, các mẫu dữ liệu trong học không giám sát chỉ cần input (đầu vào) mà không cần nhãn (đầu ra). Nó được sử dụng nhiều trong việc khám phá cấu trúc và mối quan hệ của dữ liệu. Một thuật toán điển hình là bài toán phân cụm (Clustering). Nó học cách để tìm các mẫu dữ liệu tương tự nhau và nhóm vào thành các cụm.

*\*Học tăng cường*

Học tăng cường là nhiệm vụ học thông qua thử và sai, mục tiêu của nó là đưa ra quyết định.Đây là kiểu bài toán mà máy sẽ được học một hàm mục tiêu sao cho khả năng thích nghi (được nhận “phần thưởng” nhiều nhất và ít bị “phạt” nhất) theo môi trường. Chẳng hạn rẽ trái được thưởng 1 điểm còn rẽ phải trừ 1 điểm. Việc học thích nghi theo dữ liệu môi trường nhằm tối ưu hóa một mục tiêu được đặt ra.

Khác với hoc có giám sát, trong học tăng cường không có các cặp dữ liệu vào/kết quả đúng, các hành động gần tối ưu cũng không được đánh giá đúng sai một cách tường minh.



Hình 1. : Phân loại Machine learning

## 1.3. Bài toán dự đoán ứng dụng mô hình học máy

Bài toán dự đoán (prediction) trong Machine learning được hiểu là bài toán gán nhãn, tức là quá trình từ một lượng thông tin đầu vào, thuật toán học máy có thể tính toán cho ra dự đoán về đầu ra (lable) mong muốn. Để làm được điều này nó cần nó bộ dữ liệu huấn luyện được gán nhãn trước. Như vậy học có dám sát được đề xuất để giải quyết vấn đề này. Có 2 bài toán điển hình đã được nói ở phần trên đó là phân lớp (Classification) và hồi quy (Regression). Đối với đầu ra xác định như trong bài toán dự đoán ngành nghề của chúng ta là các nhóm ngành thì thuật toán phân lớp được đề xuất. Sau đây là một số thuật toán phân lớp phổ biến được nghiên cứu trong đề tài:

### 1.3.1. Thuật toán Máy vector hỗ trợ

*\* Giới Thiệu:*

Máy vector hỗ trợ - Support Vector Machine (SVM) là một thuật toán học máy có giám sát, có thể sử dụng cho cả hai chương trình hồi quy và phân loại. Tuy nhiên thường được dùng chủ yếu và hiệu quả hơn trong bài toán phân loại. Được phát triển đầu tiên bởi Vladimir Vapnik và đồng nghiệp vào những năm 1990, SVM nhanh chóng trở thành một công cụ mạnh mẽ cho nhiều ứng dụng máy học do khả năng hiệu quả trong việc xử lý dữ liệu phức tạp.

*\* Nguyên Tắc Hoạt Động:*

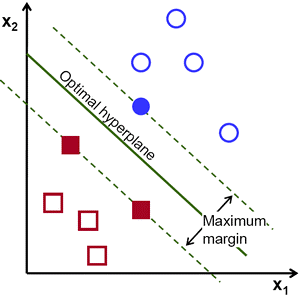
SVM hoạt động dựa trên ý tưởng tìm ra một ranh giới quyết định tối ưu giữa các lớp dữ liệu. Ranh giới này được chọn sao cho khoảng cách giữa các điểm dữ liệu gần nhất (gọi là các vector hỗ trợ) đến ranh giới là lớn nhất. Điều này có nghĩa là SVM tìm cách tối đa hóa ranh giới quyết định giữa các lớp.

*\* Hàm Mục Tiêu:*

SVM sử dụng một hàm mục tiêu để định rõ đường ranh giới. Hàm mục tiêu này bao gồm hai thành phần chính:

- Hàm chi phí (Cost Function): Đo lường sự sai lệch giữa dự đoán và thực tế. Mục tiêu là tối thiểu hóa tổng lỗi trên toàn bộ tập dữ liệu.

- Hàm biên (Margin Function): Đo lường khoảng cách từ các điểm dữ liệu đến đường ranh giới. Mục tiêu là tối đa hóa khoảng cách này.



Hình 1. : Margin trong SVM

*\* Kernel Trick:*

Một trong những đặc điểm mạnh mẽ của SVM là khả năng sử dụng Kernel Trick. Kernel Trick giúp SVM xử lý hiệu quả các dữ liệu không tuyến tính bằng cách ánh xạ chúng lên một không gian chiều cao hơn.

*\* Quá Trình Huấn Luyện:*

Huấn luyện SVM bao gồm các bước sau:

- Thu thập dữ liệu: Xác định và thu thập dữ liệu đào tạo từ các lớp khác nhau.

- Chuẩn bị dữ liệu: Tiền xử lý dữ liệu để đảm bảo chúng thích hợp cho mô hình.

- Chọn kernel và siêu tham số: Lựa chọn kernel phù hợp và điều chỉnh siêu tham số để đạt được hiệu suất tốt nhất.

- Huấn luyện mô hình: Áp dụng thuật toán SVM để xây dựng một ranh giới quyết định tối ưu.

- Đánh giá và điều chỉnh: Đánh giá hiệu suất của mô hình trên tập kiểm tra và điều chỉnh nếu cần thiết.

*\* Ưu Điểm và Nhược Điểm:*

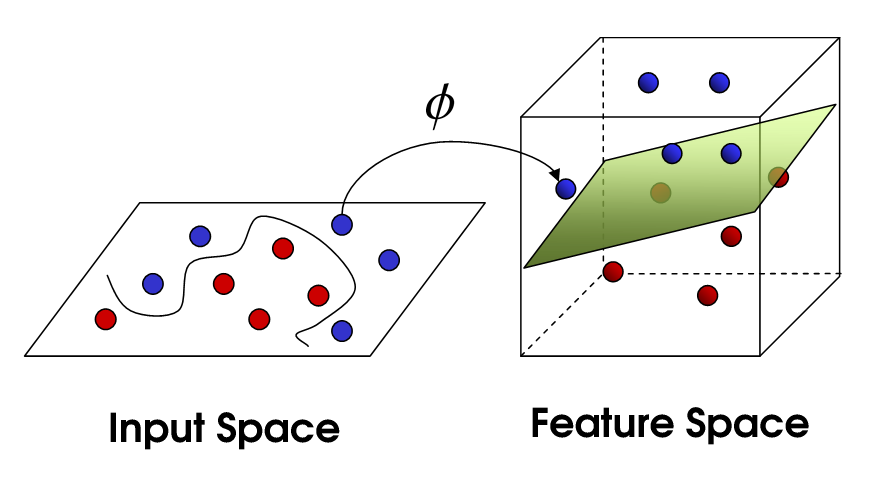
- Ưu điểm: SVM hiệu quả đối với các tập dữ liệu có kích thước lớn, đa chiều, và cả dữ liệu không tuyến tính.

- Nhược điểm: Đôi khi, việc chọn kernel và siêu tham số có thể là một thách thức, và SVM có thể đòi hỏi tài nguyên tính toán đáng kể.

Support Vector Machine là một trong những thuật toán quan trọng trong lĩnh vực máy học. Sự mạnh mẽ của nó đến từ khả năng xử lý đa dạng các loại dữ liệu và khả năng tối ưu hóa đường ranh giới quyết định. Đối với những bài toán có tập dữ liệu phức tạp, SVM vẫn là một lựa chọn đáng xem xét.

Quy trình điển hình của thuật toán: Đầu tiên mỗi mục dữ liệu là được vẽ trong không gian n chiều, trong đó n là số đối tượng và giá trị của mỗi đối tượng là giá trị của tọa độ cụ thể đó. Bước tiếp theo là phân loại bằng cách lấy siêu mặt phẳng tuyến tính (*linear* *hyper-plane*) để phân tách thành hai lớp riêng biệt. Một yếu tố làm nên hiệu quả của SVM đó là việc sử dụng thuật toán hạt nhân (***Kernel function)*** khiến cho các phương pháp chuyển không gian trở nên linh hoạt hơn. Nói một cách đơn giản nó thực hiện một số biết đổi dữ liệu phức tạp, sau đó tìm ra quá trình tách dữ liệu dựa trên các nhãn hoặc đầu ra mà chúng ra đã xác định trước.

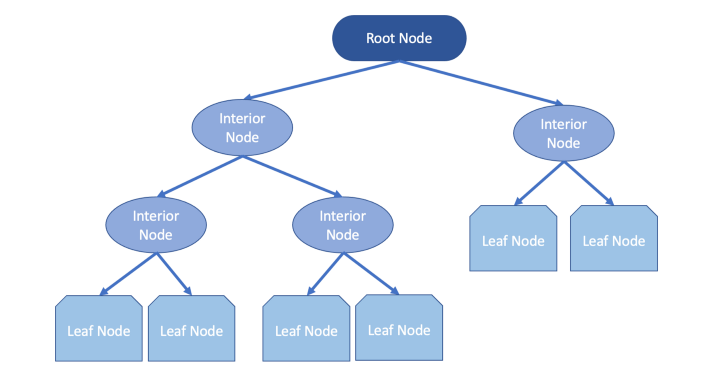
Để tính toán sự tối ưu bằng toán học, trong SVM sử dụng thuật ngữ ***Margin.*** *Margin*là khoảng cách giữa siêu phẳng đến 2 điểm dữ liệu gần nhất tương ứng với 2 phân lớp. SVM cố gắng tối ưu thuật toán bằng các tìm cách maximize giá trị *margin* này, từ đó tìm ra siêu phẳng đẹp nhất để phân 2 lớp dữ liệu.



Hình 1. : Siêu mặt phẳng trong thuật toán phân lớp SVM

### 1.3.2. Thuật toán Cây quyết định

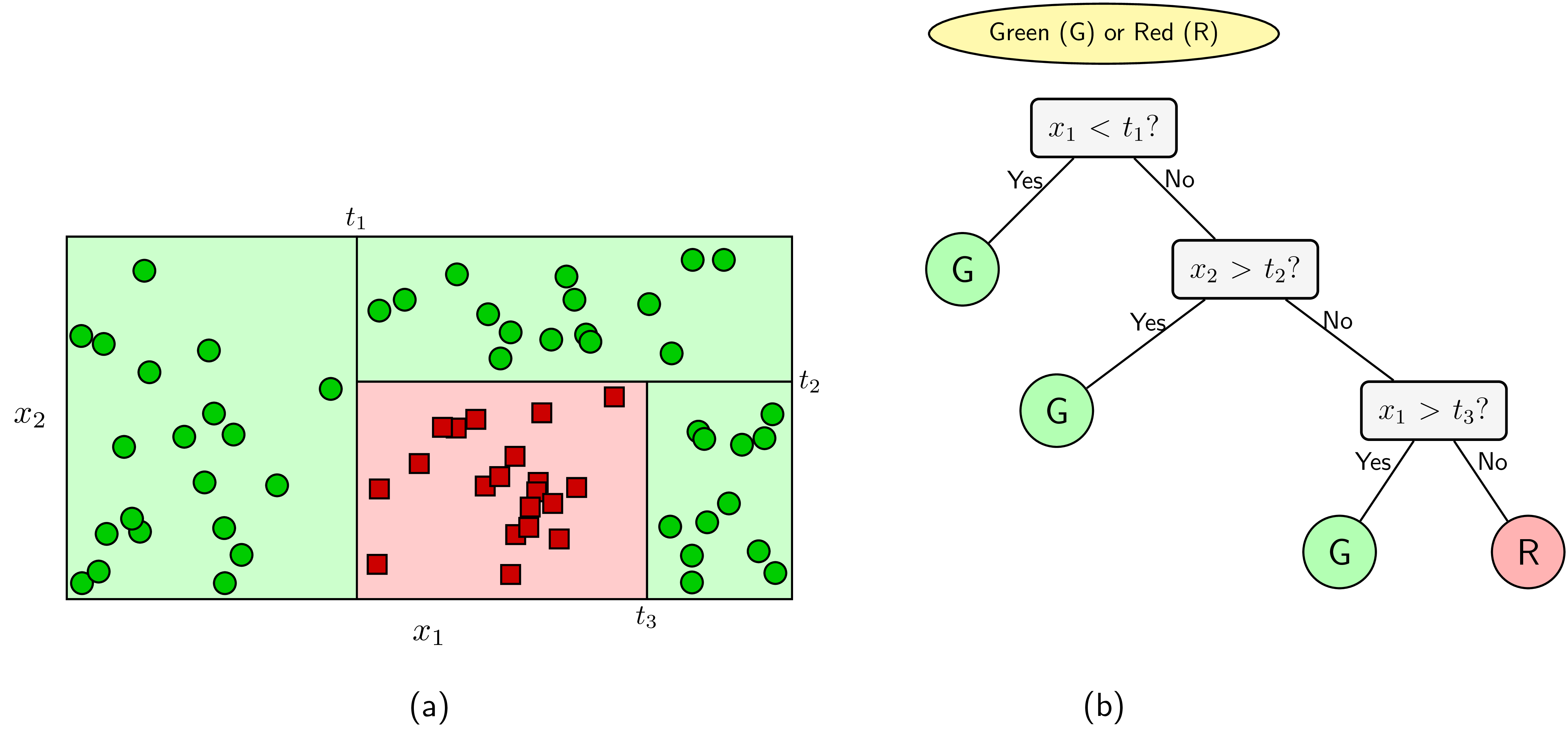
Cây quyết định (Decision Tree) [4] một trong những thuật toán phổ biến của Machine Learning thuộc nhánh học có dám sát. Decision Tree ra đời từ những năm 1975 từ một tác giả có tên Ross Quinlan. Nó sử dụng một loạt các quyết định tuần tự được thực hiện để đạt được kết quả cụ thể. Nó được dùng trong cả bài toán phân lớp và hồi quy. Cây quyết định là cây mà mỗi nút (node) biểu diễn một đặc trưng (tính chất), mỗi nhánh (branch) biểu diễn một quy luật (rule) và mỗi lá (leaf) biểu biễn một kết quả.



Hình 1. : Cấu trúc cây quyết định

Hãy nhìn vào cấu trúc cây quyết định ở trên, chúng ta có thể thấy nó bắt đầu với một nút gốc, các thuộc tính tốt nhất trở thành các nút bên trong, tức là các nút quyết định. Sau đó, các nút bên trong kiểm tra một điều kiện và thực hiện quyết định, chia không gian mẫu thành hai. Các nút lá thể hiện một phân lớp, khi bản ghi đến nút lá, thuật toán sẽ gán nhãn của lá tương ứng. Quá trình này được gọi là phân vùng đệ quy của không gian mẫu. Cây quyết định sử dụng một số hàm chi phí để chọn cách phân chia tốt nhất. Tôi đang cố gắng tìm thuộc tính / tính năng tốt nhất hoạt động tốt nhất trong việc phân loại dữ liệu đào tạo. Quá trình này được lặp lại cho đến khi đạt được nút lá và do đó, được gọi là tách nhị phân đệ quy. Khi thực hiện quy trình này, tất cả các giá trị đều được xếp thẳng hàng và cây sẽ kiểm tra các phần tách khác nhau và chọn một giá trị trả về chi phí thấp nhất, làm cho đây là một cách tiếp cận tham lam.

Xét ví dụ trên Hình a với hai class màu lục và đỏ trên không gian hai chiều. Nhiệm vụ là đi tìm ranh giới đơn giản giúp phân chia hai class này. Hay nói cách khác, đây là một bài toán classification, ta cần xây dựng một bộ phân lớp để quyết định việc một điểm dữ liệu mới thuộc vào class nào. Quan sát hình ta thấy rằng ranh giới cho hai class trong bài toán này khá đơn giản–chúng là các đường song song với các trục toạ độ. Nếu một điểm có thành phần thứ nhất, x1, nhỏ hơn ngưỡng t1, ta quyết định ngay được rằng nó thuộc class lục. Ngoài ra, nếu thành phần thứ hai, x2 lớn hơn ngưỡng t2, ta quyết định nó cũng thuộc vào class lục. Xét tiếp, nếu thành phần thứ nhất, x1, lớn hơn ngưỡng t3, ta quyết định nó thuộc vào class lục. Các điểm không thoả mãn các điều kiện trên được xếp vào class đỏ. Việc ra quyết định một điểm thuộc class nào được mô tả trên decision tree trên Hình 2b.



Hình 1. : Ví dụ về bài toán phân lớp sử dụng decision tree.

Trong decision tree, các ô màu xám, lục, đỏ trên được gọi là các node. Các node thể hiện đầu ra (màu lục và đỏ) được gọi là node lá (leaf node hoặc terminal node). Các node thể hiện câu hỏi là các non-leaf node. Non-leaf node trên cùng (câu hỏi đầu tiên) được gọi là node gốc (root node). Các non-leaf node thường có hai hoặc nhiều node con (child node). Các child node này có thể là một leaf node hoặc một non-leaf node khác. Các child node có cùng bố mẹ được gọi là sibling node. Nếu tất cả các non-leaf node chỉ có hai child node, ta nói rằng đó là một binary decision tree (cây quyết định nhị phân). Các câu hỏi trong binary decision tree đều có thể đưa được về dạng câu hỏi đúng hay sai. Các decision tree mà một leaf node có nhiều child node cũng có thể được đưa về dạng một binary decision tree. Điều này có thể đạt được vì hầu hết các câu hỏi đều có thể được đưa về dạng câu hỏi đúng sai.

Ví dụ, ta có thể xác định được tuổi của một người dựa trên nhiều câu hỏi đúng sai dạng: tuổi của bạn lớn hơn x đúng không? (Đây chính là thuật toán tìm kiếm nhị phân – binary search.)

Decision tree là một mô hình supervised learning, có thể được áp dụng vào cả hai bài toán classification và regression. Việc xây dựng một decision tree trên dữ liệu huấn luyện cho trước là việc đi xác định các câu hỏi và thứ tự của chúng. Một điểm đáng lưu ý của decision tree là nó có thể làm việc với các đặc trưng (trong các tài liệu về decision tree, các đặc trưng thường được gọi là thuộc tính – attribute) dạng categorical, thường là rời rạc và không có thứ tự. Ví dụ, mưa, nắng hay xanh, đỏ, v.v. Decision tree cũng làm việc với dữ liệu có vector đặc trưng bao gồm cả thuộc tính dạng categorical và liên tục (numeric). Một điểm đáng lưu ý nữa là decision tree ít yêu cầu việc chuẩn hoá dữ liệu.

Có 2 thuật toán cây quyết định phổ biến nhất, được phân loại dựa trên tham số đánh giá sự phân lớp, đó là:

*\* ID3 (Iterative Dichotomiser 3):* dùng Entropy Function và Information Gain

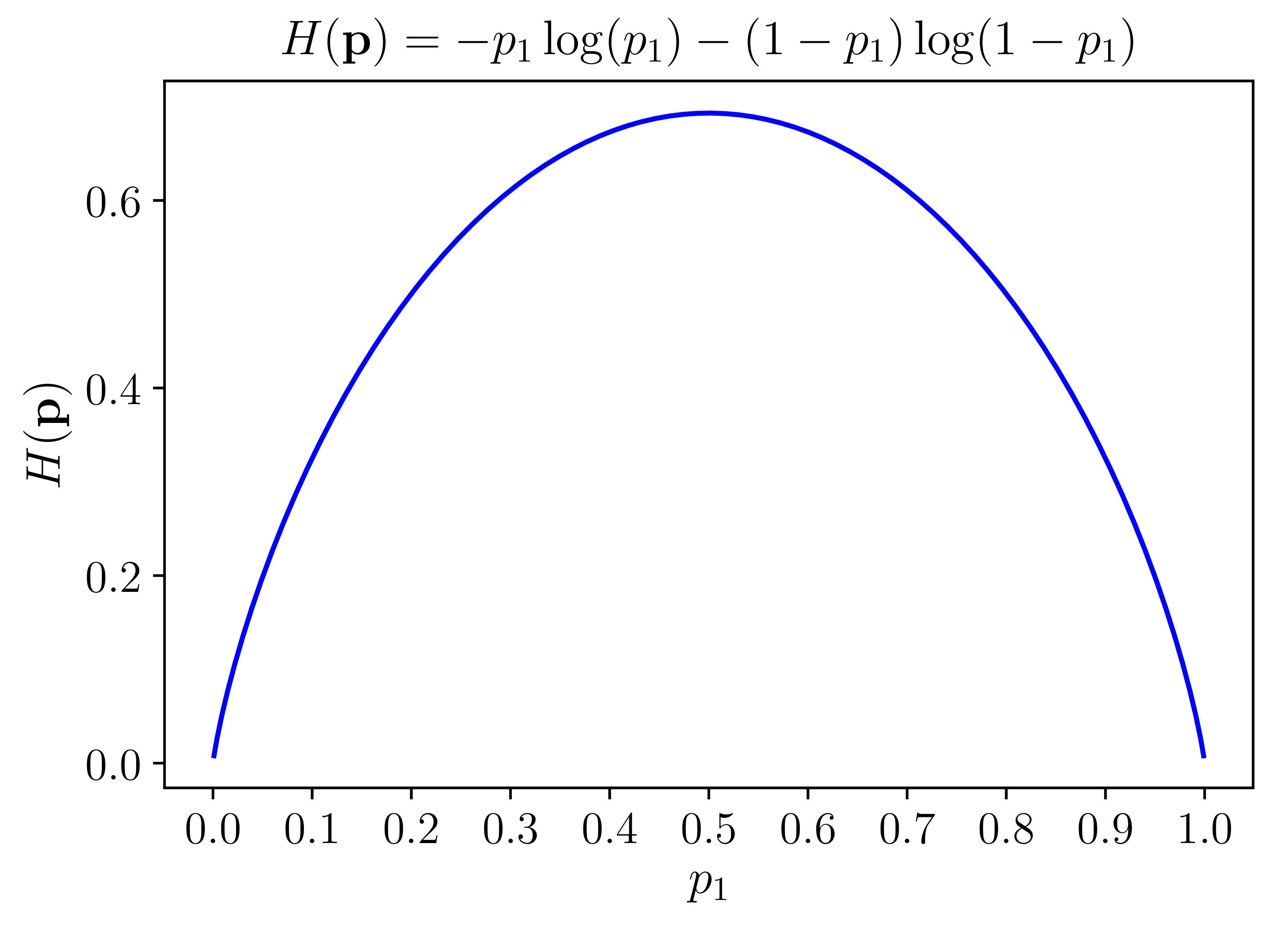
- Entropy:Được phát triển bởi Claude Shannon, entropy cung cấp cho chúng ta các thước đo về sự lộn xộn của dữ liệu. Giá trị entropy cao cho thấy khả năng dự đoán ít hơn, hãy coi entropy của một đối tượng là lượng thông tin trong đối tượng đó. Cây quyết định hoạt động để tối đa hóa độ tinh khiết của các lớp khi thực hiện tách, cung cấp sự rõ ràng hơn của các lớp trong các nút lá. Entropy được tính toán trước và sau mỗi lần tách. Nếu entropy tăng lên, một lần tách khác sẽ được thử hoặc nhánh của cây sẽ dừng lại, tức là cây hiện tại có entropy thấp nhất. Nếu entropy giảm, sự phân chia sẽ được giữ nguyên. Công thức tính toán entropy của toàn bộ tập dữ liệu:

Trong đó:

- pi: tần suất xuất hiện của các mẫu trong lớp Ci với I = {1,2, …, m}

- S: số lượng tập huấn luyện

- Si: số các mẫu S trong lớp Ci



Hình 1. : Đồ thị của hàm entropy với n=2

- Information Gain:Thông tin thu được sử dụng entropy như một thước đo tạp chất. Đó là sự khác biệt về entropy từ trước đến sau khi phân tách, và sẽ cho chúng ta một con số về độ không chắc chắn đã giảm đi bao nhiêu. Nó cũng là tiêu chí chính được sử dụng trong thuật toán cây phân loại ID3. Để tính toán mức tăng thông tin:

Trong đó:

- S là tập hợp ban đầu với thuộc tính A. Các giá trị của v tương ứng là các giá trị của thuộc tính A.

- Sv bằng tập hợp con của tập S mà có thuộc tính A mang giá trị v.

- |SV| là số phần tử của tập SV.

- |S| là số phần tử của tập S.

\* *CART* *(Classification and Regression Trees)*: dùng Gini Index (Classification)

Chỉ số Gini được đề xuất bởi nhà khoa học *Corrado Gin***,** hàm này cho chúng ta biết các nút lá trên cây “thuần khiết” như thế nào. Tạp chất gini sẽ luôn có giá trị từ 0 đến 0,5, giá trị càng cao thì nhóm càng bị rối loạn. Công thức tính tạp chất gini:

trong đó (pi) là xác suất thuộc nhóm thứ i.

Dù có độ chính xác khá cao nhưng cây quyết định tồn tại những hạn chế lớn đó là:

- Dễ xảy ra quá khớp nếu số lượng các đặc trưng để hỏi lớn. Khi độ sâu của cây quyết định không bị giới hạn thì có thể tạo ra những node lá chỉ có một vài quan sát. Những kết luận dự báo từ chúng thường chỉ đúng trên tập huấn luyện mà không đúng trên tập kiểm tra.

- Trong tình huống bộ dữ liệu có số lượng biến lớn. Một cây quyết định có độ sâu giới hạn (để giảm thiểu quá khớp) thường bỏ sót những biến quan trọng.

- Cây quyết định chỉ tạo ra một kịch bản dự báo duy nhất cho mỗi một quan sát nên nếu model có hiệu suất kém thì kết quả sẽ bị chệch.

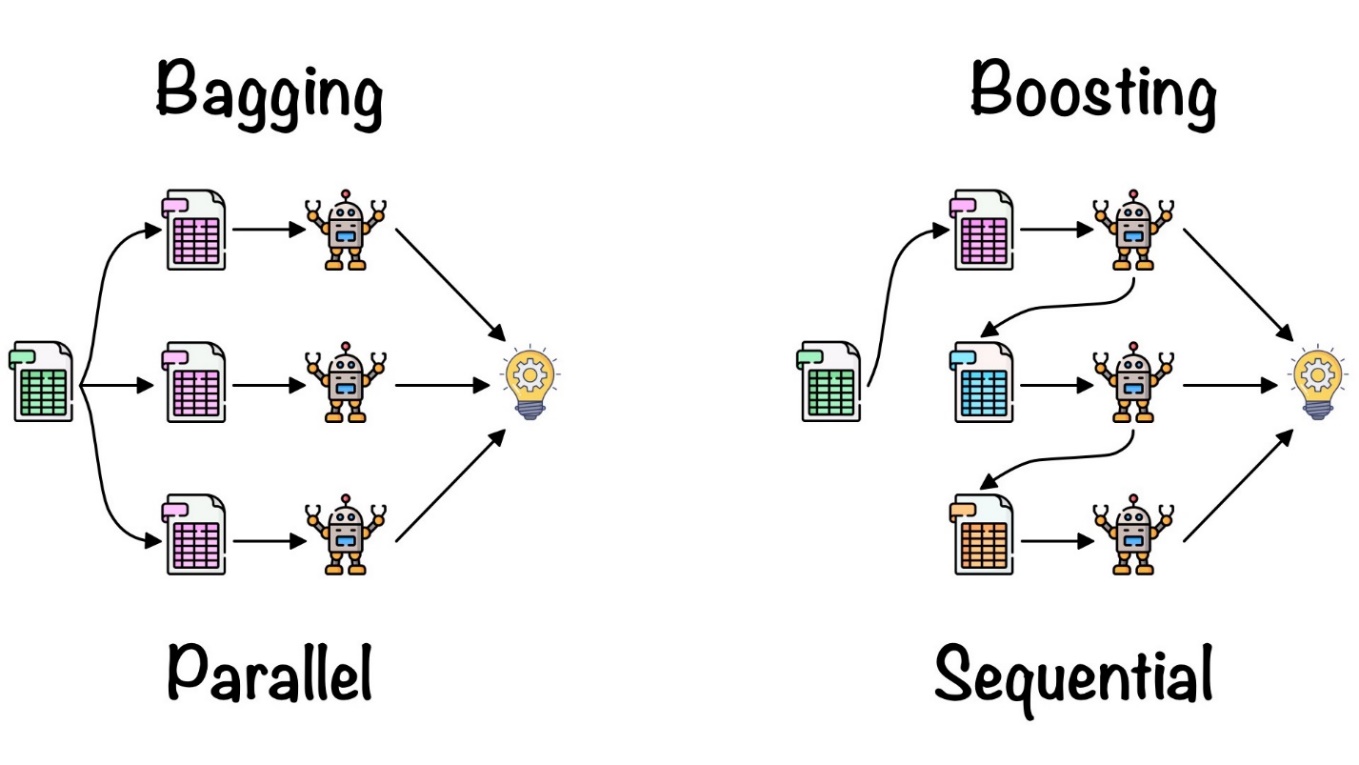
### 1.3.3. Thuật toán Học kết hợp

*\* Học kết hợp*

Học kết hợp (Ensemble learning) hiểu đơn giản đây là thuật toán kết hợp các mô hình học yếu thành mô hình học mạnh, có thể là các mô hình khác nhau hoặc áp dụng trên các phần dữ liệu khác nhau (kì vọng là độc lập với nhau). Về cơ bản, nó được phân làm 2 loại chính:

*- Bagging:* Xây dựng một lượng lớn các model (thường là cùng loại) trên những subsamples khác nhau từ tập training dataset (random sample trong 1 dataset để tạo 1 dataset mới). Những model này sẽ được train độc lập và song song với nhau nhưng đầu ra của chúng sẽ được trung bình cộng để cho ra kết quả cuối cùng.

*- Boosting:* Xây dựng một lượng lớn các model (thường là cùng loại). Mỗi model sau sẽ học cách sửa những errors của model trước (dữ liệu mà model trước dự đoán sai) -> tạo thành một chuỗi các model mà model sau sẽ tốt hơn model trước bởi trọng số được update qua mỗi model (cụ thể ở đây là trọng số của những dữ liệu dự đoán đúng sẽ không đổi, còn trọng số của những dữ liệu dự đoán sai sẽ được tăng thêm). Chúng ta sẽ lấy kết quả của model cuối cùng trong chuỗi model này làm kết quả trả về.



Hình 1. : Mô tả thuật toán Bagging và Boosting

#### 2.3.3.1. Thuật toán Rừng cây ngẫu nhiên

Rừng cây ngẫu nhiên (Random Forest) [5] là một phương pháp ensemble (kết hợp) được sử dụng để giải quyết cả bài toán phân loại và hồi quy, được phát triển bởi Breiman. Thuật toán thực hiện bằng cách kết hợp các thuật toán cây quyết đ ịnh trên tập dữ liệu với sự thay đổi có kiểm soát.

Mô hình rừng cây được huấn luyện dựa trên sự phối hợp giữa luật kết hợp (ensembling) và quá trình lấy mẫu tái lặp (boostrapping). Cụ thể thuật toán này tạo ra nhiều cây quyết định mà mỗi cây quyết định được huấn luyện dựa trên nhiều mẫu con khác nhau và kết quả dự báo là bầu cử (voting) từ toàn bộ những cây quyết định. Như vậy một kết quả dự báo được tổng hợp từ nhiều mô hình nên kết quả của chúng sẽ không bị chệch. Đồng thời kết hợp kết quả dự báo từ nhiều mô hình sẽ có phương sai nhỏ hơn so với chỉ một mô hình. Điều này giúp cho mô hình khắc phục được hiện tượng quá khớp. Chúng ta sẽ tìm hiểu về luật kết hợp và lấy mẫu tái lặp trong mô hình rừng cây, và những điểm then chốt trong quá trình huấn luyện mô hình rừng cây.

Theo mô tả của bài báo [14], thuật toán rừng ngẫu nhiên trong hồi quy được mô tả ngắn gọn theo ngôn ngữ thuật toán như sau:

\*) Giai đoạn đào tạo (Training Phase):

Cho:

- D: tập huấn luyện với n giá trị quan sát, có p đặc trưng và biến đầura.

- B: số hồi quy trong tập hợp.

Thủ tục thực hiện: Cho b = 1 đến B

1. Tạo một tập mẫu bootstrapped D𝑏∗ từ tập huấn luyện D.

2. Tạo cây hồi quy bằng cách sử dụng mẫu bootstrapped D𝑏∗.

Đối với một nút t cho trước,

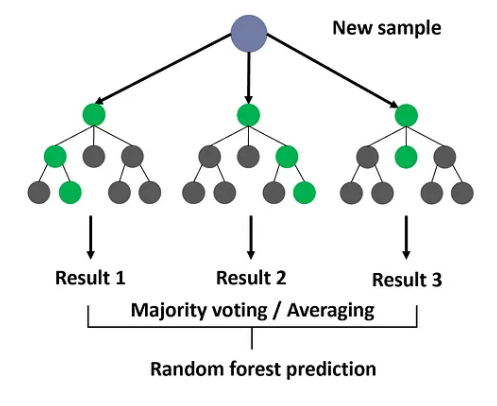
(i) Lấy mẫu ngẫu nhiên một số đặc trưng (features) từ bộ đặc trưng đầy đủ.

(ii) Tìm quy tắc tách tốt nhất bằng cách sử dụng tập hợp con đặc trưng trong mẫu ngẫu nhiên.

(iii) Tách nút t thành hai nút con bằng cách sử dụng quy tắc tách tốt nhất. Lặp lại bước (i) - (iii) cho đến khi đáp ứng các quy tắc dừng.

3. Lấy một bộ hồi quy đã được huấn luyện Rb.

\*) Giai đoạn thử nghiệm (Test Phase): Đối với trường hợp áp dụng bộ thử nghiệm x, giá trị dự đoán được ước tính bởi các bộ hồi quy B và được đưa ra dưới dạng công thức:



Hình 1. : Mô tả thuật toán Rừng cây ngẫu nhiên

\* Ưu Điểm:

- Ổn Định và Hiệu Quả: Random Forest thường cho kết quả ổn định và hiệu quả, đặc biệt là trên các tập dữ liệu lớn và đa dạng.

- Xử Lý Dữ Liệu Lớn: Có khả năng xử lý dữ liệu lớn và có nhiều đặc trưng mà không mất đi tính hiệu suất.

- Dễ Điều Chỉnh Tham Số: Có tham số có thể được điều chỉnh để tối ưu hóa hiệu suất trên dữ liệu cụ thể của bạn.

\* Nhược Điểm:

- Khó Để Hiểu và Giải Thích: Sự phức tạp của nhiều cây có thể làm cho Random Forest trở nên khó hiểu và khó giải thích so với các mô hình đơn giản hơn.

- Chiếm nhiều tài nguyên: Yêu cầu nhiều tài nguyên tính toán và bộ nhớ so với một số mô hình đơn giản hơn.

#### 2.3.3.2. Thuật toán XGBoost

XGBoost (eXtreme Gradient Boosting) [6] là một thuật toán học tổng hợp (Ensemble learning) máy dựa trên cây quyết định sử dụng tăng cường độ dốc. Nó được coi là thuật toán học máy tốt nhất hiện nay cho các dữ liệu dạng bảng có cấu trúc vừa, nhỏ. Để hiểu rõ về Xgboost, tôi sẽ trình bài những khái niệm chung nhất liên quan tới nó.

Đây là loại Boosting tiến hành đánh trọng số cho các mô hình mới được thêm vào dựa trên phương pháp tối ưu Gradient Descent [11]

Nhằm giải quyết bài toán tối ưu:

Trong đó: L: hàm loss, y: lable, : điểm tin cậy (trọng số) của mô hình yếu thứ n, : mô hình yếu thứ n.

Thay vì cố gằng quét tìm tất cả các giá trị , để tìm nghiệm tối ưu toàn cục như ở Bagging - một công việc tốn nhiều thời gian và tài nguyên, chúng ta sẽ cố gắng tìm các giá trị nghiệm cục bộ sau khi thêm mỗi một mô hình mới vào chuỗi mô hình với mong muốn dần đi đến nghiệm toàn cục.

Với

nếu chúng ta coi chuỗi các mô hình boosting là một hàm số W, thì mỗi hàm learner có thể coi là một tham số *w*. Đến đây, để cực tiểu hóa hàm loss L (y, W), chúng ta áp dụng Gradient Descent:

Đến đây ta có thể thấy mối quan hệ:

Giá trị này còn được gọi là pseudo-residuals

Tóm lại, chúng ta có thể tóm tắt quá trình triển khai thuật toán như sau:

1. Khởi tạo giá trị pseudo-residuals là bằng nhau cho từng điểm dữ liệu.

2. Tại vòng lặp thứ i

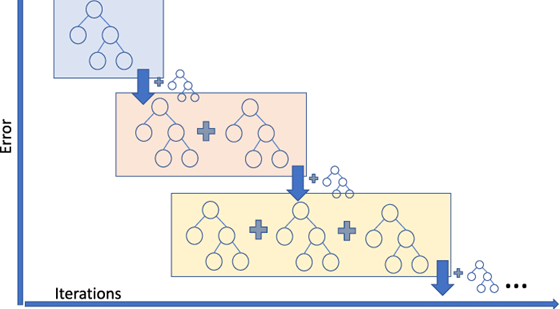
- Train mô hình mới được thêm vào để phù hợp vào giá trị của pseudo-residuals đã có

- Tính toán giá trị của mô hình vừa train

- Cập nhật model chính

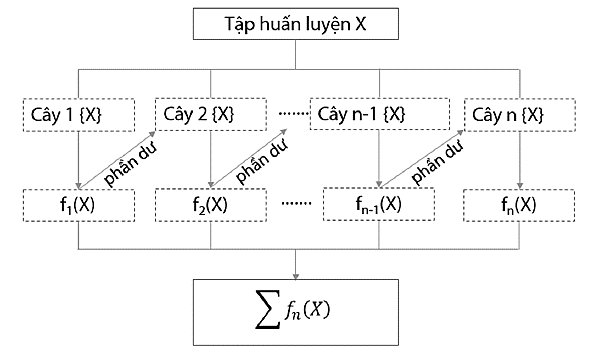
- Cuối cùng, tính toán giá trị pseudo-residuals  để làm lable cho mô hình tiếp theo

3. Sau đó lặp lại với vòng lặp i + 1.



Hình 1. : Mô tả thuật toán Gradient Boosting

Thuật toán eXtreme Gradient Boosting viết tắt là XGBoost là một thuật toán mới được giới thiệu bởi Tianqi Chen và Carlos Guestrin vào năm 2016. XGBoost nổi lên như một công cụ mạnh mẽ trong nhiều lĩnh vực phân loại và chiến thắng trong nhiều cuộc thi phân lớp do Kaggle tổ chức. XGBoost được nâng cấp từ thuật toán Gradient Tree Boosting (GTB) do Fiedman và cộng sự đề xuất năm 2000. Nguyên lý cơ bản được sử dụng trong thuật toán GTB là việc kết hợp các mô hình học tập cơ bản yếu (tức là có độ sai số cao) thành một cây mô hình học tập mạnh hơn theo kiểu tuần tự. Chúng ta có thể xem quá trình học của thuật toán GTB như minh học trong hình 1.



Hình 1. : Mô hình thuật toán Gradient Tree Boosting

Giả sử rằng chúng ta có một tập huấn luyện có N mẫu với thông số đầu ra xác định . Tại vòng lặp đầu tiên, một cây học tập bất kỳ được tạo ra và ước lượng được các giá trị đầu ra là . Các thông số ước lượng này sẽ có sai khác với giá trị chính xác một lượng giá trị được gọi là phần dư . Phần dư có thể hiểu là biểu thị cho sai số của mô hình. Muốn mô hình học tập tốt hơn thì chúng ta phải giảm giá trị phần dư này đi. Để thực hiện điều này, cây học tập thứ 2 sẽ được xây dựng nhằm mục đích ước lượng các giá trị của phần dư đó (không phải là giá trị y). Tương tự như trên, khi ước lượng phần dư , cây học tập thứ 2 sẽ ước lượng được giá trị tạo ra phần dư . Để ước lượng phần dư , ta lại tiếp tục tạo ra cây học tập thứ 3. Quá trình lặp cứ liên tục như vậy. Cuối cùng thì giá trị ước lượng sẽ là .

Để nâng cao hiệu suất làm việc của mô hình GTB, trong mô hình XGBoost một thành phần được thêm vào trong hàm mất mát (loss funcition). Lúc này hàm mất mát của mô hình XGBoost có dạng:

(2.1)

Trong đó các tham số của mô hình được huấn luyện kí hiệu là ; L là hàm loos function, là thành phần được thêm vào thường được gọi là regularization nhằm đánh giá độ phức tạp của mô hình. Việc thêm vào thành phần regularization giúp làm hài hòa các tham số thu được của mô hình học và tránh hiện tượng mô hình quá khớp (overfitting). Theo kinh nghiệm, việc sử dụng hàm mục tiêu được chuẩn hóa như trong công thức 2.1 sẽ giúp mô hình được lựa chọn có xu hướng sử dụng các hàm đơn giản và có thể dự đoán. Mô hình càng đơn giản sẽ cho phép tránh hiện tượng quá khớp càng tốt. Do dựa vào mô hình học tập dạng cây, giá trị ước đoán cuối cùng sẽ là:

Hàm mất mát ở vòng lặp thứ t có dạng:

Giá trị ước lượng cho đầu ra ở vòng lặp thứ t, được tính như sau:

Giá trị regularization có thể sử dụng công thức sau để xác định:

Trong đó là độ phức tạp của các lá trong cây quyết định; T là số lá trong cây quyết định; là hệ số phóng đại hàm phạt, w là vectơ điểm số cho các lá. Khi đó, phân tích bậc 2 Taylor sẽ được sử dụng trong hàm mất mát ở thuật toán XGBoost thay thế cho phân tích bậc nhất được sử dụng trong thuật toán GTB. Giả thiết rằng hàm tối ưu cho quá trình học tập là hàm MSE, khi đó hàm mục tiêu sẽ được viết thành:

(2.6)

Trong công thức 2.6, các hằng số đã được loại bỏ. q(.) là hàm số dừng để gán dữ liệu cho lá tương ứng, và h là đạo hàm bậc nhất và bậc hai của hàm mất mát MSE. Trong công thức 2.6 hàm mất mát được xác định bằng tổng của các giá trị mất mát cho từng mẫu. bởi vì mỗi mẫu chỉ tương ứng với 1 lá cho nên hàm mất mát có thể xác định bằng tổng các giá trị mất mát cho mỗi lá. Do đó, ta có:

(2.7)

Trong đó đại diện cho tất cả các mẫu ở lá . Lúc này bài toán tối ưu của hàm mất mát có thể chuyển thành bài toán tìm giá trị nhỏ nhất của một hàm bậc 2. Nói cách khác, sau khi phân chia một nút nhất định trong cây ra quyết định, sự thay đổi hiệu suất của mô hình được cải thiện sau khi thực hiện sự phân chia nút này, sự phân chia đó sẽ được chấp nhận, ngược lại việc tách nút sẽ dừng lại. Bên cạnh đó, nhờ việc sử dụng thành phần regularization sẽ giúp cho mô hình tránh được hiện tượng quá khớp.

Giải thuật XGBoost được mô tả như sau:

- Đầu vào: Một tập hợp các dữ liệu huấn luyện. Dữ liệu này bao gồm các thuộc tính mô tả một đối tượng hay một tình huống nào đó, và một giá trị phân loại.

- Đầu ra: Cây quyết định có khả năng phân lớp tập dữ liệu rèn luyện, và hy vọng rằng cây đó có khả năng phân lớp đúng cho cả các dữ liệu chưa gặp trong tương lai.

Xgboost là một giải thuật được base trên gradient boosting với base model là Decision Tree, tuy nhiên kèm theo đó là những cải tiến to lớn về mặt tối ưu thuật toán, về sự kết hợp hoàn hảo giữa sức mạnh phần mềm và phần cứng, giúp đạt được những kết quả vượt trội cả về thời gian training cũng như bộ nhớ sử dụng. Thuật toán XGBoost được phát triển như một dự án nghiên cứu tại Đại học Washington. Tianqi Chen và Carlos Guestrin đã trình bày bài báo của họ tại Hội nghị SIGKDD (Special Interest Group on Knowledge Discovery and Data )vào năm 2016 và khiến thế giới Machine Learning bùng nổ. Kể từ khi được giới thiệu, thuật toán này không chỉ được ghi nhận là đã chiến thắng nhiều cuộc thi Kaggle mà còn là động lực cho một số ứng dụng tiên tiến trong ngành. Mã nguồn mở với ~350 contributors và ~3,600 commits trên Gihub, XGBoost cho thấy những khả năng ứng dụng đáng kinh ngạc của mình như :

- Song song hóa: XGBoost tiếp cận quá trình xây dựng cây tuần tự bằng cách sử dụng thực hiện song song. Điều này thực hiện được do tính chất có thể hoán đổi cho nhau của các vòng lặp được sử dụng để xây dựng người học cơ sở; vòng lặp bên ngoài liệt kê các nút lá của cây và vòng lặp bên trong thứ hai tính toán các đối tượng địa lý. Việc lồng các vòng lặp hạn chế việc song song hóa bởi vì nếu không hoàn thành vòng lặp bên trong (yêu cầu tính toán nhiều hơn trong hai vòng lặp), vòng lặp bên ngoài không thể được bắt đầu. Do đó, để cải thiện thời gian chạy, thứ tự các vòng được thay đổi cho nhau bằng cách sử dụng khởi tạo thông qua quét toàn cục tất cả các phiên bản và sắp xếp bằng cách sử dụng các luồng song song. Kỹ thuật này cải thiện hiệu suất thuật toán bằng cách bù đắp bất kỳ chi phí song song nào trong tính toán.

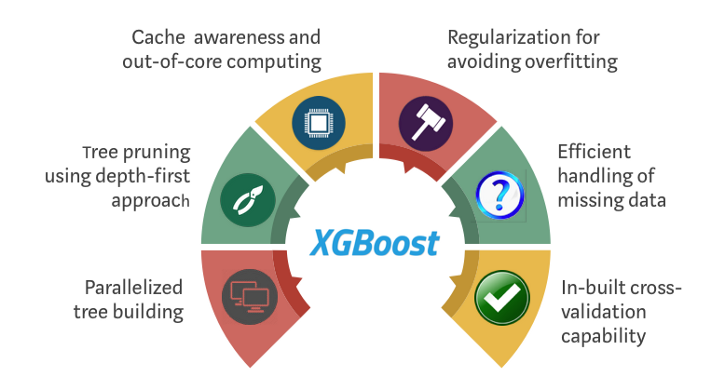
- Tỉa cây: Tỉa là một kỹ thuật học máy để giảm kích thước của cây hồi quy bằng cách thay thế các nút mà không góp phần nâng cao phân loại trên lá. Ý tưởng của việc cắt tỉa cây hồi quy là để ngăn chặn việc trang bị quá nhiều dữ liệu huấn luyện. XGBoost tạo các nút (còn được gọi là phần tách) lên đến ‘max\_depth’ được chỉ định và bắt đầu cắt bỏ từ phía sau cho đến khi tổn thất dưới một ngưỡng. Cách tiếp cận 'theo chiều sâu' này cải thiện đáng kể hiệu suất tính toán.

- Tối ưu hóa phần cứng: Thuật toán này được thiết kế để sử dụng hiệu quả tài nguyên phần cứng. Điều này được thực hiện nhờ nhận thức bộ nhớ cache bằng cách phân bổ bộ đệm nội bộ trong mỗi luồng để lưu trữ thống kê gradient. Các cải tiến khác như tính toán 'ngoài lõi' tối ưu hóa dung lượng ổ đĩa khả dụng trong khi xử lý các khung dữ liệu lớn không vừa với bộ nhớ.

- Nhận thức về độ thưa thớt: XGBoost tự nhiên thừa nhận các tính năng thưa thớt cho đầu vào bằng cách tự động tìm giá trị còn thiếu tốt nhất tùy thuộc vào sự mất mát trong quá trình đào tạo và xử lý các loại mẫu độ thưa thớt khác nhau trong dữ liệu hiệu quả hơn.

- Phác thảo lượng tử có trọng số: XGBoost sử dụng thuật toán Phác thảo lượng tử có trọng số phân tán để tìm hiệu quả các điểm phân tách tối ưu giữa các tập dữ liệu có trọng số.

- Xác thực chéo: Thuật toán đi kèm với phương pháp xác thực chéo được tích hợp sẵn ở mỗi lần lặp, loại bỏ nhu cầu lập trình rõ ràng tìm kiếm này và chỉ định số lần lặp thúc đẩy chính xác được yêu cầu trong một lần chạy.



Hình 1. : Ưu điểm của thuật toán Xgboost

Tuy nhiên XGBoost vẫn có một số nhược điểm như sau:

- XGBoost sử dụng các phương pháp tối ưu hóa mang lại hiệu quả tốt hơn nhưng các phương pháp tối ưu của nó dẫn đến rất tốn kém về mặt chi phí tính toán.

- XGBoost cố gắng khớp nối với các dữ liệu được đem vào huấn luyện nó làm tăng độ chính xác nhưng cũng dẫn tới dễ bị overfitting đối với model.

## 1.4. Đánh giá mô hình phân lớp

Khi xây dựng một mô hình Machine Learning, chúng ta cần một phép đánh giá để xem mô hình sử dụng có hiệu quả không và để so sánh khả năng của các mô hình.

### 1.4.1. Các phương pháp đánh giá mô hình học máy

*\* Phương pháp train - test dataset*

Đây là phương pháp đơn giản nhất để đánh giá một ML model. Từ tập dữ liệu ban đầu, ta chia thành 2 phần, gọi là trainset và testset theo tỉ lệ nhất định (thường là 7:3, 8:2 hoặc thậm chí 9:1 tùy theo kích thước của tập và đặc trưng của tập dữ liệu). Sau đó, tiến hành train model trên train set rồi sử dụng model đã train đó để dự đoán trên tập test set. Dựa trên kết quả của dự đoán để đưa ra đánh giá chất lượng của model.

Ưu điểm của phương pháp này là nhanh. Nó sẽ phù hợp để áp dụng khi bài toán của bạn đáp ứng ít nhất 1 trong 2 tiêu chí sau:

- Tập dữ liệu có kích thước lớn (hàng triệu mẫu) và có cơ sở để tin rằng cả 2 phần dữ liệu đều đại diện đầy đủ cho tất cả các khía cạnh của vấn đề cần dự đoán (để chắc chắn hơn về điều này, ta có thể xáo trộn ngẫu nhiên tập dữ liệu trước khi chia)

- Thuật toán train của model rất lâu để hội tụ.

Nếu điều kiện thứ 2 không thỏa mãn mà ta vẫn sử dụng phương pháp này thì sẽ gặp phải vấn đề high variance. Tức là khi 2 tập train set và test set chứa những đại diện khác nhau của vấn đề cần dự đoán thì kết quả đánh giá trên tập testset không thể hiện đúng chất lượng của model.

*\* Phương pháp k - fold cross – validation*

Cross-validation là phương pháp mở rộng của phương pháp bên trên để hạn chế được vấn đề high variance.

Các bước tiến hành của nó như sau:

- Xáo trộn dữ liệu một cách ngẫu nhiên.

- Chia tập dữ liệu ban đầu thành k phần (k=5,10,...), mỗi phần gọi là một fold. - train model trên k-1 fold và đánh giá trên fold còn lại.

- Lặp lại k lần bước bên trên để mỗi fold trong tập dữ liệu đều có cơ hội trở thành test set.

- Sau khi toàn bộ quá trình kết thúc ta sẽ có k kết quả đánh giá khác nhau, kêt quả cuối cùng sẽ được tổng hợp dựa vào trung bình (mean) và độ lệch chuẩn (standard deviation) của k kết quả đó.

Phương pháp này cho kết quả đánh giá tin cậy hơn so với phương pháp sử dụng train-test set bởi vì nó được train và đánh giá nhiều lần trên các tập dữ liệu khác nhau. Việc lựa chọn k cũng cần phải xem xét sao cho kích thước của mỗi fold đủ lớn để dữ liệu trong mỗi fold mang tính đại diện cao về mặt thống kê của toàn bộ dữ liệu. Thực nghiệm cho thấy k=5 hoặc k=10 là lựa chọn tốt nhất cho hầu hết các trường hợp.

Nhận xét:

Nói chung, k-fold cross-validation là phương pháp tốt nhất cho việc đánh giá hiệu năng của một ML model trong hầu hết mọi trường hợp.

Sử dụng train-test set trong trường hợp thuật toán train mất nhiều thời gian để hội tụ và số lượng mẫu của dữ liệu rất lớn.

Lời khuyên hợp lý nhất là hãy thử nghiệm nhiều lần và tìm ra phương pháp phù hợp với bài toán sao cho nhanh nhất có thể. Phương pháp được coi là phù hợp khi nó kết quả đánh giá của nó đáp ứng đúng (hoặc gần đúng) yêu cầu bài toán đặt ra ban đầu.

### 1.4.2. Đánh giá mô hình phân lớp

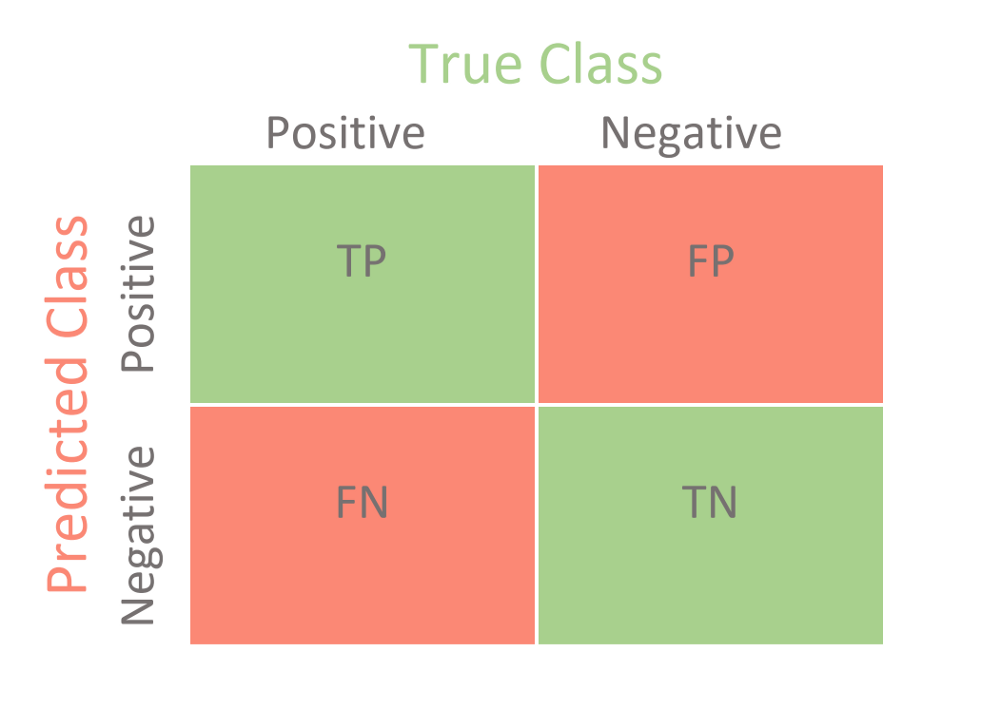
Hiệu năng của mô hình thường đánh giá qua tập dữ liệu kiểm thử (testset). Cụ thể, giả sử đầu ra của mô hình khi đầu vào là tập kiểm thử được mô tả bởi vector y\_pred - là vector dự đoán đầu ra với mỗi phần tử là class được dự đoán của một điểm dữ liệu trong tập kiểm thử. Ta cần so sánh giữa vector dự đoán y\_pred này với vector class thật của dữ liệu, được mô tả bởi vector y\_true.

Có rất nhiều cách đánh giá một mô hình phân lớp. Tuỳ vào những bài toán khác nhau mà chúng ta sử dụng các phương pháp khác nhau. Các phương pháp thường được sử dụng là: accuracy score và confusion matrix [15]

Cách đơn giản và hay được sử dụng nhất là accuracy (độ chính xác). Cách đánh giá này đơn giản tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử.

Cách tính sử dụng accuracy như ở trên chỉ cho chúng ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là confusion matrix.

Về cơ bản, confusion matrix là một phương pháp đánh giá kết quả của những bài toán phân loại với việc xem xét cả những chỉ số về độ chính xác và độ bao quát của các dự đoán cho từng lớp. Một confusion matrix gồm 4 chỉ số sau đối với mỗi lớp phân loại:



Hình 1. : Mô tả confusion matrix

Để đơn giản hóa, ta sẽ sử dụng lại bài toán về chẩn đoán ung thư để giải thích 4 chỉ số này. Trong bài toán chuẩn đoán ung thư ta có 2 lớp: lớp bị ung thư được chuẩn đoán Positive và lớp không bị ung thư được chuẩn đoán là Negative:

- TP (True Positive): Số lượng dự đoán chính xác. Là khi mô hình dự đoán đúng một người bị ung thư.

- TN (True Negative): Số lương dự đoán chính xác một cách gián tiếp. Là khi mô hình dự đoán đúng một người không bị ung thư, tức là việc không chọn trường hợp bị ung thư là chính xác.

- FP (False Positive - Type 1 Error): Số lượng các dự đoán sai lệch. Là khi mô hình dự đoán một người bị ung thư và người đó hoàn toàn khỏe mạnh.

- FN (False Negative - Type 2 Error): Số lượng các dự đoán sai lệch một cách gián tiếp. Là khi mô hình dự đoán một người không bị ung thư nhưng người đó bị ung thư, tức là việc không chọn trường hợp bị ung thư là sai

Từ 4 chỉ số này, ta các chỉ số để đánh giá mức độ tin cậy của một mô hình:

- Accuracy: Đây là độ đo của bài toán phân loại mà đơn giản nhất, tính toán bằng cách lấy số dự đoán đúng chia cho toàn bộ các dự đoán.

Nhược điểm của cách đánh giá này là chỉ cho ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất hay dữ liệu của lớp nào thường bị phân loại nhầm nhất vào các lớp khác.

### 1.4.3. Đánh giá mô hình học máy với dữ liệu mất cân bằng

Mất cân bằng dữ liệu là một trong những hiện tượng phổ biến của bài toán phân loại nhị phân (binary classification) như spam email, phát hiện gian lận, dự báo vỡ nợ, chuẩn đoán bệnh lý, … Trong trường hợp tỷ lệ dữ liệu giữa 2 classes là 50:50 thì được coi là cân bằng. Khi có sự khác biệt trong phân phối giữa 2 classes, chẳng hạn 60:40 thì dữ liệu có hiện tượng mất cân bằng.

Hầu hết các bộ dữ liệu đều khó đạt được trạng thái cân bằng mà luôn có sự khác biệt về tỷ lệ giữa 2 classes. Đối với những trường hợp dữ liệu mất cân bằng nhẹ như tỷ lệ 60:40 thì sẽ không ảnh hưởng đáng kể tới khả năng dự báo của mô hình.

Tuy nhiên nếu hiện tượng mất cân bằng nghiêm trọng xảy ra, chẳng hạn như tỷ lệ 90:10 sẽ thường dẫn tới ngộ nhận chất lượng mô hình. Khi đó thước đo đánh giá mô hình là độ chính xác (accuracy) có thể đạt được rất cao mà không cần tới mô hình. Ví dụ, một dự báo ngẫu nhiên đưa ra tất cả đều là nhóm đa số thì độ chính xác đã đạt được là 90%. Do đó không nên lựa chọn độ chính xác làm chỉ số đánh giá mô hình để tránh lạc quan sai lầm về chất lượng.

Trong trường hợp mẫu mất cân bằng nghiêm trọng ta cần phải thay đổi chỉ số đánh giá để đưa ra kết quả hợp lý hơn.

Ngoài ra, mất cân bằng dữ liệu nghiêm trọng thường dẫn tới dự báo kém chính xác trên nhóm thiểu số. Bởi đa phần kết quả dự báo ra thường thiên về 1 nhóm là nhóm đa số và rất kém trên nhóm thiểu số. Trong khi tầm quan trọng của việc dự báo được chính xác một mẫu thuộc nhóm thiểu số lớn hơn nhiều so với dự báo mẫu thuộc nhóm đa số. Để cải thiện kết quả dự báo chúng ta cần những điều chỉnh thích hợp để mô hình đạt được một độ chính xác cao trên nhóm thiểu số.

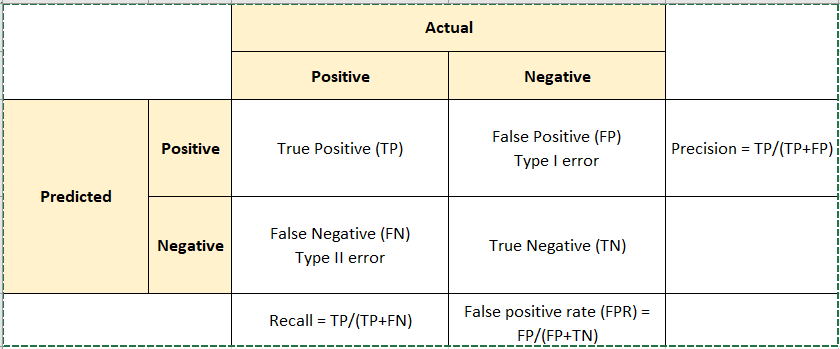
Một số giải pháp cho dữ liệu mất cân bằng như sau:

*\* Thay đổi metric*

Như đã giải thích ở trên, khi hiện tượng mất cân bằng dữ liệu nghiêm trọng xảy ra thì việc sử dụng độ chính xác làm thước đo đánh giá mô hình thường không hiệu quả bởi hầu hết chúng đều đạt độ chính xác rất cao. Một mô hình ngẫu nhiên dự báo toàn bộ là nhãn thuộc nhóm đa số cũng sẽ mang lại kết quả gần bằng 100%. Khi đó ta có thể cân nhắc tới một số metrics thay thế như precision, recall, f1-score, .... Các chỉ số này sẽ không quá lớn để dẫn tới ngộ nhận độ chính xác, đồng thời chúng tập trung hơn vào việc đánh giá độ chính xác trên nhóm thiểu số, nhóm mà chúng ta muốn dự báo chính xác hơn so với nhóm đa số.

- Precision: Trong tất cả các dự đoán Positive được đưa ra, bao nhiêu dự đoán là chính xác? Chỉ số này được tính theo công thức:

- Recall: Trong tất cả các trường hợp Positive, bao nhiêu trường hợp đã được dự đoán chính xác? Chỉ số này được tính theo công thức:



Hình 1. : Mô tả chỉ số Precision và Recall

- F1-Score: Trung bình điều hòa giữa Precision và Recall. Đây là chỉ số thay thế lý tưởng cho accuracy khi mô hình có tỷ lệ mất cân bằng mẫu cao.

*\* Giảm lượng mẫu*

Giảm lượng mẫu - Under sampling là việc ta giảm số lượng các quan sát của nhóm đa số để nó trở nên cân bằng với số quan sát của nhóm thiểu số. Ưu điểm của under sampling là làm cân bằng mẫu một cách nhanh chóng, dễ dàng tiến hành thực hiện mà không cần đến thuật toán giả lập mẫu.

Tuy nhiên nhược điểm của nó là kích thước mẫu sẽ bị giảm đáng kể. Gỉa sử nhóm thiểu số có kích thước là 500, như vậy để tạo ra sự cân bằng mẫu giữa nhóm đa số và thiểu số sẽ cần giảm kích thước mẫu của nhóm đa số từ 10000 về 500. Tổng kích thước tập huấn luyện sau under sampling là 1000 và chiếm gần 1/10 kích thước tập huấn luyện ban đầu. Tập huấn luyện mới khá nhỏ, không đại diện cho phân phối của toàn bộ tập dữ liệu và thường dễ dẫn tới hiện tượng overfitting.

Do đó trong một số phương án, chúng ta có thể không nhất thiết lựa chọn sao cho tỷ lệ mẫu giữa nhóm đa số: nhóm thiểu số là 50:50 mà có thể giảm dần xuống về 80:20, 70:30 hoặc 60:40 và tìm ra phương án nào mang lại hiệu quả dự báo tốt nhất trên tập kiểm tra.

*\* Tăng lượng mẫu*

Over sampling là các phương pháp giúp giải quyết hiện tượng mất cân bằng mẫu bằng cách gia tăng kích thước mẫu thuộc nhóm thiểu số bằng các kĩ thuật khác nhau. Có 2 phương pháp chính để thực hiện over sampling đó là:

- Lựa chọn mẫu có tái lập.

- Mô phỏng mẫu mới dựa trên tổng hợp của các mẫu cũ: SMOTE (Synthetic Minority Over-sampling) và ADASYN (Adaptive synthetic sampling)

***SMOTE***, viết tắt của Synthetic Minority Over-sampling Technique, là một phương pháp được sử dụng để cân bằng dữ liệu trong bài toán phân loại, đặc biệt là khi có sự mất cân bằng giữa các lớp (minority class và majority class). Phương pháp này được thiết kế để tạo ra các mẫu tổng hợp (synthetic samples) từ minority class để giảm thiểu hiện tượng mất cân bằng.

Dưới đây là cách SMOTE hoạt động:

1. Chọn một mẫu từ minority class: Lựa chọn một mẫu ngẫu nhiên từ minority class để tạo mẫu tổng hợp.

2. Chọn k-nearest neighbors: Để tạo mẫu tổng hợp, chọn k mẫu láng giềng gần nhất của mẫu đã chọn trong không gian đặc trưng.

3. Tính toán khoảng cách: Đo khoảng cách giữa mẫu đã chọn và mỗi mẫu láng giềng.

4. Tạo mẫu tổng hợp: Với mỗi mẫu láng giềng, tính tỉ lệ alpha (thường nằm trong khoảng từ 0 đến 1) và sử dụng nó để tạo một mẫu tổng hợp mới bằng cách kết hợp đặc trưng của mẫu đã chọn và đặc trưng của mẫu láng giềng.

5. Thêm mẫu tổng hợp vào tập dữ liệu: Thêm các mẫu tổng hợp vào tập dữ liệu của minority class.

Quy trình này được lặp lại cho mỗi mẫu trong minority class để tạo ra một lượng lớn mẫu tổng hợp và cân bằng sự mất cân bằng trong tập dữ liệu.

Lưu ý rằng, trong một số trường hợp, việc sử dụng SMOTE có thể dẫn đến tình trạng overfitting, vì mẫu tổng hợp có thể không phản ánh chính xác đặc tính của minority class. Do đó, việc lựa chọn thí cụm mẫu tổng hợp và điều chỉnh các tham số là quan trọng khi triển khai phương pháp này.

***ADASYN*** (Adaptive Synthetic Sampling) là một phương pháp cân bằng dữ liệu giống như SMOTE, được thiết kế để giảm hiện tượng mất cân bằng giữa các lớp trong bài toán phân loại. Tương tự như SMOTE, ADASYN tạo ra các mẫu tổng hợp từ minority class để làm tăng cường dữ liệu cho lớp thiểu số. Tuy nhiên, điểm khác biệt quan trọng của ADASYN so với SMOTE là khả năng điều chỉnh mức độ tăng cường dữ liệu cho từng mẫu dựa trên độ chệch cụ thể của mỗi mẫu.

Dưới đây là cách ADASYN hoạt động:

1. Chọn một mẫu từ minority class: Lựa chọn một mẫu ngẫu nhiên từ minority class để tạo mẫu tổng hợp.

2. Chọn k-nearest neighbors: Tìm k mẫu láng giềng gần nhất của mẫu đã chọn trong không gian đặc trưng.

3. Tính toán tỷ lệ alpha: Tính tỷ lệ alpha cho mỗi mẫu láng giềng, trong đó alpha được đặc trưng bởi tỷ lệ giữa số lượng mẫu thuộc majority class và minority class trong k láng giềng.

4. Tạo mẫu tổng hợp: Với mỗi mẫu láng giềng, tính toán mẫu tổng hợp mới bằng cách kết hợp đặc trưng của mẫu đã chọn và đặc trưng của mẫu láng giềng, theo tỷ lệ alpha đã tính.

5. Thêm mẫu tổng hợp vào tập dữ liệu: Thêm các mẫu tổng hợp vào tập dữ liệu của minority class.

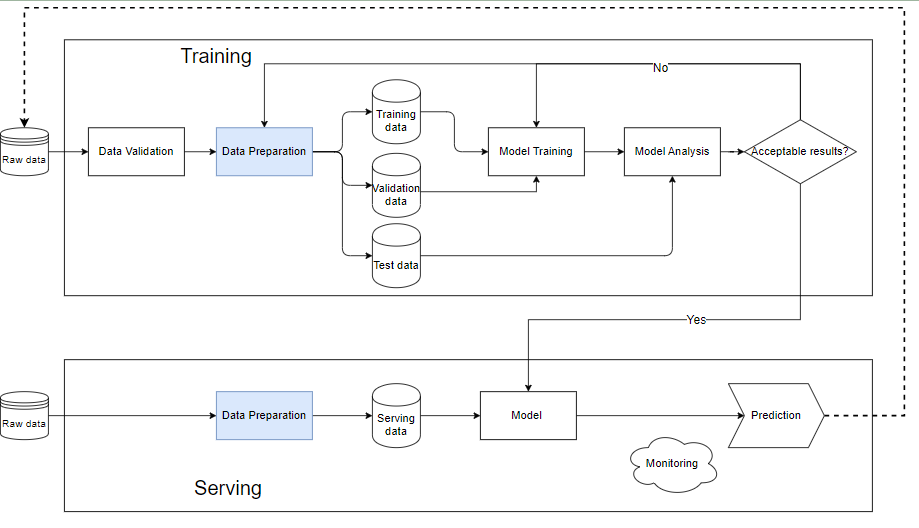
Quy trình này được lặp lại cho mỗi mẫu trong minority class để tạo ra một lượng lớn mẫu tổng hợp và giảm mất cân bằng trong tập dữ liệu. Điều này giúp mô hình học được mối quan hệ phức tạp hơn giữa các lớp và cải thiện khả năng dự đoán trên lớp thiểu số.

## 1.5. Quy trình triển khai một dự án học máy

Hệ thống machine learning thường bao gồm rất nhiều các thành phần nhỏ như xử lý dữ liệu, huấn luyện mô hình, đánh giá mô hình, dự đoán với dữ liệu mới, v.v. Nếu không xây dựng một pipeline hoàn chỉnh với từng thành phần tách biệt rõ ràng, sẽ có rất nhiều vấn đề xảy ra.

Điểm đặc biệt của machine learning khi so sánh với các hệ thống phần mềm thông thường là rất khó để viết tests cho từng thành phần. Code của bạn có thể có rất nhiều lỗi nhưng hệ thống vẫn chạy bình thường. Dữ liệu có thể thay đổi phân phối, một đặc trưng nào đó có thể có lỗi khi thu thập, metric đánh giá có thể có lỗi, hệ thống của bạn vẫn có thể âm thầm chạy. Và thật nguy hiểm nếu hệ thống vẫn chạy và cho ra kết quả ngẫu nhiên. Việc tách cả hệ thống ra từng thành phần rồi ghép lại thành một pipeline giúp bạn có thể viết các logic kiểm tra cho từng bước và chia nhỏ pipeline ra để tìm thành phần gây lỗi mỗi khi có điều bất thường.

Ngoài sự thuận tiện trong việc rà soát lỗi, có pipeline với các thành phần riêng biệt còn giúp việc làm việc nhóm trở nên dễ dàng hơn. Nếu có nhóm lớn, ta có thể chia ra thành từng nhóm nhỏ: một nhóm chuyên làm sạch dữ liệu, một nhóm chuyên tạo các đặc trưng, nhóm khác đi xây dựng và huấn luyện mô hình và một nhóm khác tập trung vào đánh giá và theo dõi hoạt động của mô hình. Những khối công việc này nếu được tách nhỏ và chuyên biệt sẽ giúp các nhóm đi sâu vào cải thiện chất lượng của từng khối mà không lo đến việc làm hỏng code của nhóm khác. Ngoài ra, mỗi khi có một ý tưởng mới hoặc mô hình mới, bạn cũng sẽ dễ dàng thay đổi thành phần đó để kiểm nghiệm mà không phải đi xây dựng lại toàn bộ hệ thống. Việc này hữu ích không chỉ trong nhóm lớn mà với khi làm việc độc lập. Bạn có 5-10 ý tưởng về việc xây dựng mô hình khác nhau, bạn không cần phải xây dựng lại cả 5-10 khối code khác nhau cho từng mô hình mà chỉ cần thay thế phần tử “mô hình” trong cả pipelie.



Hình 1. : Luồng thực hiện dự án học máy

Có hai khối chính là “Training” và “Serving” thể hiện luồng dữ liệu trong quá trình huấn luyện và khi chạy thực tế.

\* Pha “Training”:

“Data Validation” là bước kiểm định dữ liệu. Bước này ít được nhắc tới trong các tài liệu nhưng lại rất quan trọng. Tại bước này, dữ liệu cần được kiểm tra xem có tương tự như dữ liệu hiện có trong cơ sở dữ liệu hay không. Lưu ý rằng trong các bài toán thực tế, dữ liệu mới thường xuyên được sinh ra, và mô hình cần thường xuyên được cập nhật dựa trên dữ liệu mới này để “bắt kịp xu thế”. Việc kiểm tra dữ liệu có tương tự không có thể được xác định dựa trên các thông số thống kê. Chẳng hạn với bài toán dự đoán một người dùng có click vào một quảng cáo hay không, nếu tỉ lệ click/không click hiện tại chỉ là 1% mà trong dữ liệu mới, tỉ lệ này là 50% thì nhiều khả năng dữ liệu mới chứa nhiều thông tin spam. Nếu để mô hình huấn luyện trên dữ liệu này, rất nhiều khả năng mô hình của bạn sẽ có kết quả tệ.

Qua bước kiểm định, dữ liệu thô được làm sạch, tạo đặc trưng trong bước “Data Prepration” sau đó được tách ra thành tập huấn luyện (training data) và tập kiểm định (validation data). Ngoài ra, bước này cũng có thể tạo ra một tập dữ liệu thứ ba là tập kiểm tra (test data) để xác định chất lượng mô hình sau quá trình huấn luyện. Trong nhiều trường hợp với ít dữ liệu huấn luyện, người ta thường dùng chính tập kiểm định làm tập kiểm tra. Khi đó cần thận trọng với việc tập kiểm định này bị overfit sau khi được dùng đi dùng lại để ra quyết định thay đổi mô hình.

Dữ liệu huấn luyện được đưa vào khối “Model Training” để huấn luyện mô hình. Dữ liệu kiểm định được sử dụng trong quá trình huấn luyện này để tinh chỉnh tham số (hyperparameter tuning).

Mô hình sau khi huấn luyện được đưa vào khối “Model Analysis” để phân tích chất lượng. Phần phân tích chất lượng này có thể bao gồm nhiều phần: metrics trên tập kiểm tra có tốt không, metrics trên từng phần dữ liệu có tốt tương đương nhau không, mô hình có dự đoán thiên lệch không, có hiện tượng rò rỉ dữ liệu không, tầm quan trọng của từng đặc trưng như thế nào, tốc độ dự đoán và nhiều phần phân tích khác tùy thuộc vào mỗi bài toán.

Nếu kết quả nhận được trong bước “Model Analysis” thỏa mãn các yêu cầu của bài toán, mô hình này sẽ được dùng cho việc chạy dữ liệu thực tế ở bước “Serving”. Nếu không, ta cần quay lại bước xây dựng mô hình, xây dựng thêm đặc trưng hoặc thậm chí phải lấy thêm dữ liệu.

\* Pha “Serving”:

Khi đã huấn luyện được một mô hình, ta có thể đưa nó vào chạy với dữ liệu mới thời gian thực. Trên thực tế, không bao giờ một mô hình mới được áp dụng ngay cho toàn bộ dữ liệu thực tế trong thời gian thực. Thay vào đó, mô hình mới chỉ được áp dụng lên một phần nhỏ của dữ liệu và so sánh chất lượng với mô hình hiện tại. Nếu chất lượng trên phần nhỏ dữ liệu này chấp nhận được, tỉ lệ dữ liệu mà mô hình mới dự đoán được nâng dần lên tới khi nó được chạy trên toàn bộ dữ liệu.

Trong pha này, bước “Data Preparation” để làm sạch và tạo dữ liệu phải giống hệt như những gì đã được làm để xây dựng mô hình trong pha “Training”. Ta cần đảm bảo dữ liệu đầu vào của mô hình có những tính chất giống hệt như những gì nó được nhìn thấy trong quá trình huấn luyện. Toàn bộ cách làm sạch, cách chuẩn hóa, mã hóa đặc trưng cần phải được thực hiện tương tự.

Kết quả mà mô hình dự đoán có thể ảnh hưởng trực tiếp đến hành vi người dùng. Phản ứng của người dùng, trong rất nhiều trường hợp, được dùng để huấn luyện những mô hình tiếp theo. Lấy ví dụ bài toán gợi ý sản phẩm, mô hình có tác dụng xếp hạng các sản phẩm theo thứ tự ưa thích của người dùng. Người dùng thường sẽ có những hành động phụ thuộc mạnh vào kết quả xếp hạng đó. Toàn bộ các trường thông tin khi mô hinh dự đoán cũng như phản ứng của người dùng sẽ được lưu lại để làm dữ liệu huấn luyện cho các mô hình tiếp theo. Như vậy, có một quan hệ khăng khít giữa chất lượng dữ liệu và chất lượng mô hình. Việc có một thành phần giúp theo dõi dự đoán của mô hình cũng như phản ứng của người dùng là cực kỳ quan trọng trong nhiều bài toán. Đó là thành phần “Monitoring” trong hình.

Mỗi bài toán khác nhau cần có một hệ thống theo dõi và cảnh báo khác nhau. Thường thì với bất kể dữ liệu đầu vào như thế nào, mô hình vẫn sẽ cho ra một kết quả dự đoán, có thể là ngẫu nhiên. Nếu các kết quả này không được theo dõi và báo động một cách kỹ lưỡng dẫn đến việc chất lượng mô hình cũng như phản ứng của người dùng thay đổi một cách đột ngột thì uy tín cũng như doanh thu của công ty sẽ bị ảnh hưởng trầm trọng.

# Chương 2: Xây dựng ứng dụng hỗ trợ gợi ý địa điểm du lịch tại Huế

## 2.1. Môi trường và công cụ

### 2.1.1. Cài đặt Anaconda

Anaconda là một nền tảng phân phối miễn phí của ngôn ngữ lập trình Python và R cho tính toán khoa học (khoa học dữ liệu, machine learning, xử lý dữ liệu lớn, phân tích dự đoán, v.v.), nhằm mục đích đơn giản hóa việc quản lý và triển khai gói. Nó có trên cả Windows, MacOS và Linux.

Anaconda chứa tất cả các gói (công cụ) phổ biến nhất mà một nhà khoa học dữ liệu cần. Các package trong Anaconda được quản lý bởi trình quản lý riêng của nền tảng này là **conda.** Ta thường dùng conda để tạo môi trường cô lập các dự án của mình, nhằm sử dụng các phiên bản Python khác nhau hoặc các phiên bản package khác nhau, cũng như dùng nó để cài đặt, gỡ cài đặt và cập nhật các package riêng trong từng dự án.

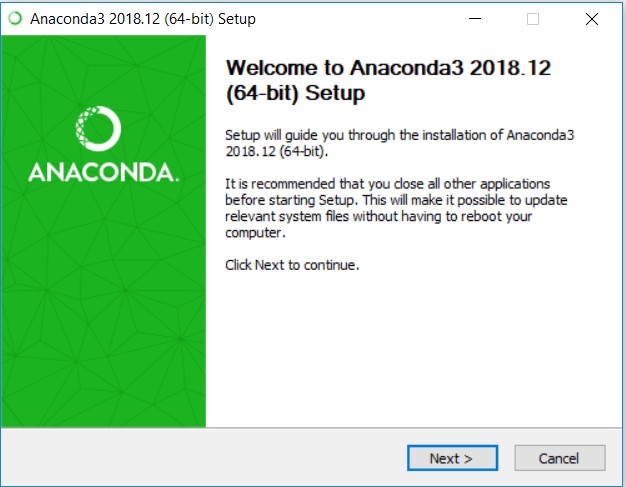
\* Tải Anaconda:

Đối với hệ điều hành window:<https://repo.anaconda.com/archive/Anaconda3-2023.02-Windows-x86_64.exe>

\* Cài đặt Anaconda:

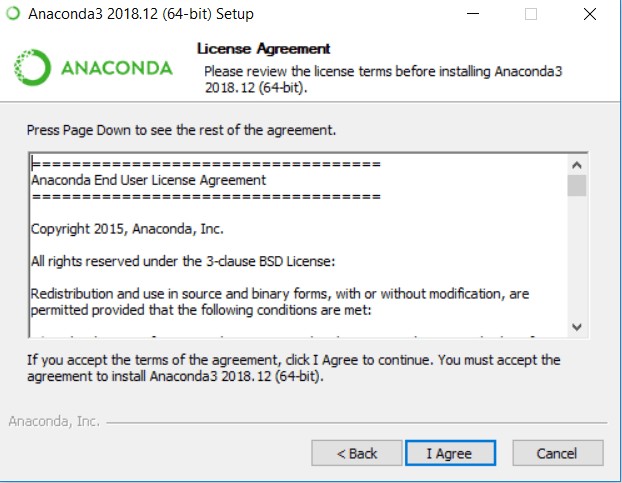
Đối với hệ điều hành window:

Click vào [**Anaconda3-2023.02-Windows-x86\_64.exe**](https://repo.anaconda.com/archive/Anaconda3-2020.02-Windows-x86_64.exe) để mở file



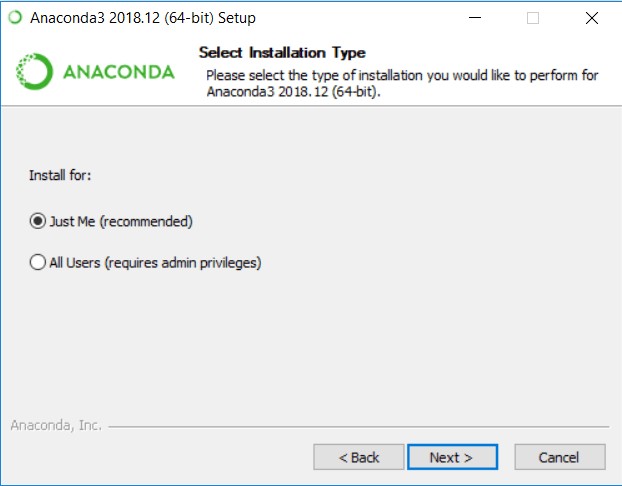
Hình 2. : Cài đặt Anaconda bước 1

-> Click Next



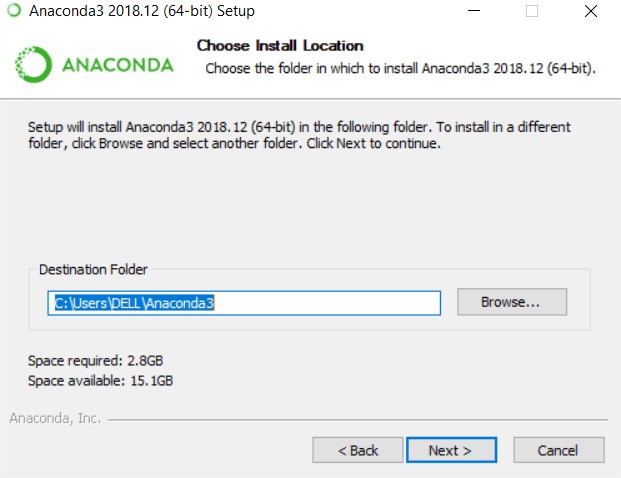
Hình 2. : Cài đặt Anaconda bước 2

-> Click I Agree



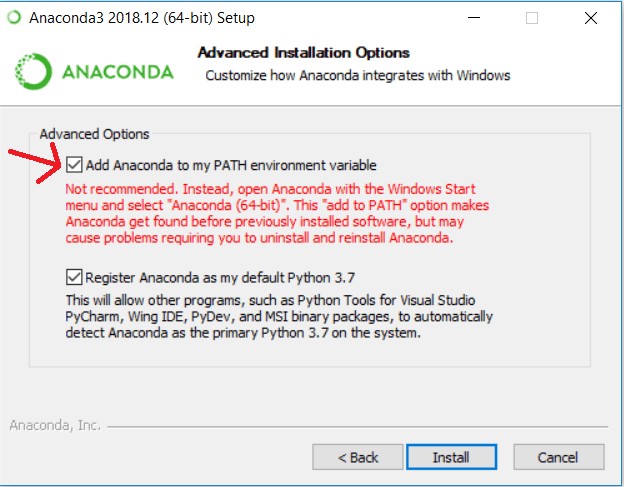
Hình 2. : Cài đặt Anaconda bước 3

-> Click Next



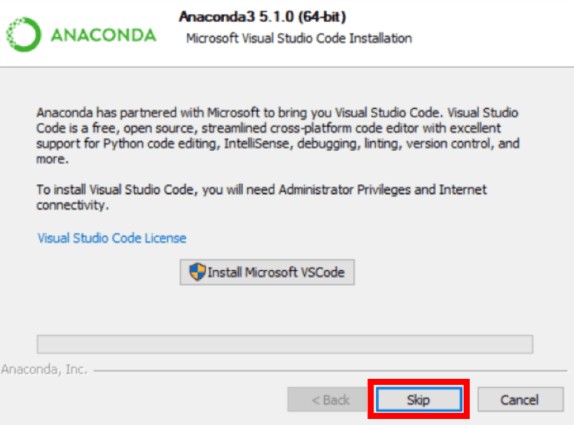
Hình 2. : Cài đặt Anaconda bước 4

-> có thể chọn như mục cài đặt khác bằng việc click Browse… và Click Next



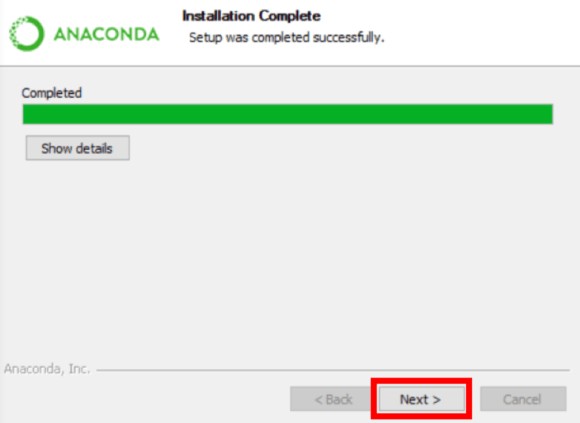
Hình 2. : Cài đặt Anaconda bước 5

-> Click vào ô vuông gần dòng Add Anaconda to my PATH environment variable.



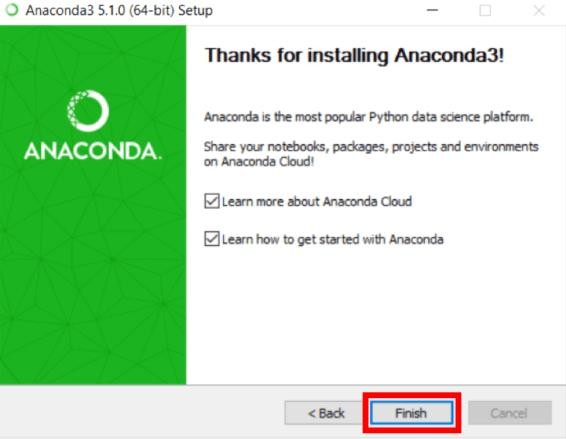
Hình 2. : Cài đặt Anaconda bước 6

-> Click next



Hình 2. : Cài đặt Anaconda bước 7

-> Click Skip



Hình 2. : Cài đặt Anaconda bước 8

-> Click Finish.

### 2.1.2. Cài đặt Python

Python là một ngôn ngữ lập trình hướng đối tượng rất thông dụng dùng để viết các tiện ích hệ thống và các đoạn mã trên Internet. Nó cũng được sử dụng như ngôn ngữ kết dính đóng vai trò tích hợp C và C++. Được tạo ra bởi Guido van Rossum tại Amsterdam năm 1990. Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý. Python là ngôn ngữ có hình thức khá đơn giản và rõ ràng, do đó tạo nên sự dễ dàng tiếp cận cho những lập trình viên mới bắt đầu. Tính năng của Python:

- Ngôn ngữ lập trình đơn giản, dễ học: Python có cú pháp rất đơn giản, rõ ràng. Nó dễ đọc và viết hơn rất nhiều khi so sánh với những ngôn ngữ lập trình khác như C++, Java, C#. Python làm cho việc lập trình trở nên thú vị, cho phép tập trung vào những giải pháp chứ không phải cú pháp

- Miễn phí, mã nguồn mở: Chúng ta có thể tự do sử dụng và phân phối Python, thậm chí là dùng cho mục đích thương mại. Vì là mã nguồn mở, không những có thể sử dụng các phần mềm, chương trình được viết trong Python mà còn có thể thay đổi mã nguồn của nó. Python có một cộng đồng rộng lớn, không ngừng cải thiện nó mỗi lần cập nhật

- Khả năng di chuyển: Các chương trình Python có thể di chuyển từ nền tảng này sang nền tảng khác và chạy nó mà không có bất kỳ thay đổi nào. Nó chạy liền mạch trên hầu hết tất cả các nền tảng như Windows, macOS, Linux.

- Thư viện tiêu chuẩn lớn để giải quyết những tác vụ phổ biến: Python có một số lượng lớn thư viện tiêu chuẩn giúp cho công việc lập trình trở nên dễ thở hơn rất nhiều, đơn giản vì không phải tự viết tất cả code.

- Khả năng mở rộng và có thể nhúng: Giả sử một ứng dụng đòi hỏi sự phức tạp rất lớn, bạn có thể dễ dàng kết hợp các phần code bằng C, C++ và những ngôn ngữ khác (có thể gọi được từ C) vào code Python. Điều này sẽ cung cấp cho ứng dụng của bạn những tính năng tốt hơn cũng như khả năng scripting mà những ngôn ngữ lập trình khác khó có thể làm được.

\*Cài đặt python

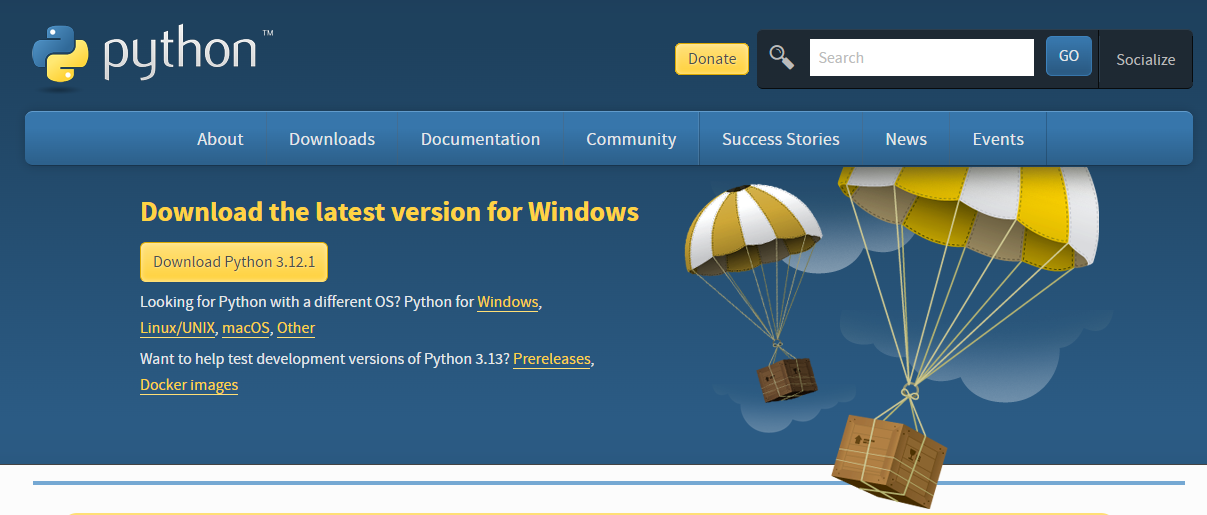
- Đối với hệ điều hành Windows:

Bước 1: Tải Python tại đây: [https://www.python.org/downloads/](https://quantrimang.com/url?q=aHR0cHM6Ly93d3cucHl0aG9uLm9yZy9kb3dubG9hZHMv), chọn phiên bản bạn cần, trong ứng dụng này sử dụng chọn Python 3.11

Bước 2: Nhấp đúp vào file vừa tải về để cài đặt. Tại đây có 2 tùy chọn, bạn chọn một cái để cài.

- Install Now: Mặc định cài Python vào ổ C, cài sẵn IDLE (cung cấp giao diện đồ họa để làm việc với Python), pip và tài liệu, tạo shortcut.

- Customize installation: Cho phép bạn chọn vị trí cài và tính năng cần thiết.



Hình 2. : Cài đặt Python

### 2.1.3. Cài đặt Scikit-learn

Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python.Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, dimensionality reduction.

Thư viện được cấp phép bản quyền chuẩn FreeBSD và chạy được trên nhiều nền tảng Linux. Scikit-learn được sử dụng như một tài liệu để học tập.

Để cài đặt scikit-learn trước tiên phải cài thư viện SciPy (Scientific Python). Những thành phần gồm:

- Numpy: Gói thư viện xử lý dãy số và ma trận nhiều chiều

- SciPy: Gói các hàm tính toán logic khoa học

- [Matplotlib](https://codelearn.io/sharing/ve-bieu-do-voi-thu-vien-matplotlib-p1): Biểu diễn dữ liệu dưới dạng đồ thị 2 chiều, 3 chiều

- IPython: Notebook dùng để tương tác trực quan với Python

- SymPy: Gói thư viện các kí tự toán học

- Pandas: Xử lý, phân tích dữ liệu dưới dạng bảng

Những thư viện mở rộng của SciPy thường được đặt tên dạng SciKits. Như thư viện này là gói các lớp, hàm sử dụng trong thuật toán học máy thì được đặt tên là scikit-learn.

Scikit-learn hỗ trợ mạnh mẽ trong việc xây dựng các sản phẩm. Nghĩa là thư viện này tập trung sâu trong việc xây dựng các yếu tố: dễ sử dụng, dễ code, dễ tham khảo, dễ làm việc, hiệu quả cao.



Hình 2. : Logo thư viện Scikit-learn

\* Cài đặt

Nếu bạn đã có một bản cài đặt numpy và scipy đang hoạt động, thì cách dễ nhất để cài đặt scikit-learning là sử dụng pip

*pip install -U scikit-learning*

hoặc conda:

*conda install -c conda-forge scikit-learning*

### 2.1.4. Cài đặt XGBoost

Cách để cài đặt thư viện XGBoost.

- Sử dụng pip để cài đặt:

*pip install xgboost*

Để cập nhật thư viện, sử dụng lệnh sau:

*pip install --upgrade xgboost*

- Sử dụng conda

*conda install -c anaconda py-xgboost*

- Biên dịch từ mã nguồn

Sử dụng cách này nếu muốn cài đặt phiên bản mới nhất của xgboost.

*git clone --recursive https://github.com/dmlc/xgboost*

*cd xgboost*

*cp make/minimum.mk ./config.mk*

*make -j8*

*cd python-package*

*sudo python setup.py install*

### 2.1.5. Cài đặt Flask

Flask là một web frameworks, nó thuộc loại micro-framework được xây dựng bằng ngôn ngữ lập trình Python. Flask cho phép bạn xây dựng các ứng dụng web từ đơn giản tới phức tạp. Nó có thể xây dựng các api nhỏ, ứng dụng web chẳng hạn như các trang web, blog, trang wiki hoặc một website dựa theo thời gian hay thậm chí là một trang web thương mại. Flask cung cấp cho bạn công cụ, các thư viện và các công nghệ hỗ trợ bạn làm những công việc trên.

Flask là một micro-framework. Điều này có nghĩa Flask là một môi trường độc lập, ít sử dụng các thư viện khác bên ngoài. Do vậy, Flask có ưu điểm là nhẹ, có rất ít lỗi do ít bị phụ thuộc cũng như dễ dàng phát hiện và xử lý các lỗi bảo mật.



Hình 2. : Thư viện Flask

Framework là Flask được chọn bởi vì những ưu điểm của nó:

- Không yêu cầu tool hay thư viện cụ thể nào.

- Cung cấp một lõi chức năng "súc tích" nhất cho ứng dụng web nhưng người dùng có thể mở rộng bất cứ lúc nào

- Người dùng có thể tập trung xây dựng web application ngay từ đầu trong một khoảng thời gian rất ngắn và có thể phát triển quy mô của ứng dụng tùy theo yêu cầu.

- Không bị bó buộc bởi bộ khung cồng kềnh, không gặp bất cứ khó khăn nào khi cấu hình hay tổ chức ứng dụng.

- Cực kỳ linh hoạt, tối giản, dễ tìm hiểu và sử dụng, định tuyến dễ dàng, rất dễ mở rộng.

*\*Cài đặt*

Flask chỉ hỗ trợ cho các phiên bản Python 3.4 hoặc mới hơn; hoặc python 2.7 và Pypy. Nếu bạn đã cài Python và thiết lập môi trường thành công. Bạn có thể sử dụng command sau đây để cài Flask:

*Pip install Flask*

Ngoài ra, trong dự án còn dùng một số thư viện khác như:

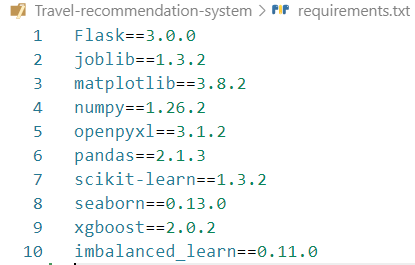
Joblib: Lưu và load model

Matplotlib, seaborn: Để vẽ đồ thị

imbalanced\_learn: thư viện xử lí dữ liệu mất cân bằng

Pandas, openpyxl: Xử lí dữ liệu bảng, thao tác với file excel

Thông tin các thư viện cần dùng và các phiên bản được lưu trong file *requirements.txt* của thư mục dự án.



Hình 2. : Tệp requirements.txt chứa thông tin các thư viện

## 2.3. Tạo tập dữ liệu

### 2.3.1. Thu thập dữ liệu

Thu thập dữ liệu là một trong những nhiệm vụ chính và quan trọng nhất của bất kỳ dự án máy học nào. Vì vậy, các thuật toán hiệu quả và độ chính xác phụ thuộc vào tính đúng đắn và chất lượng của dữ liệu. Dữ liệu được thu thập theo nhiều cách.

Trong đề tài, phần lớn dữ liệu được thu thập từ khảo sát các thu khách du lịch về thông tin các nhân của họ và đánh giá về địa điểm du lịch tại Huế.

Cụ thế quá trình thu thập như sau:

- Gửi mẫu khảo sát cho các công ty lữ hành tại Huế. Có yêu cầu cung cấp chính xác thông tin. Mẫu khảo sát này được tổng hợp online và gồm những nội dung sau:

+ Họ tên: Thông tin Họ và tên của người được khảo sát

+ Thông tin nghề nghiệp, mức thu nhập

+ Thông tin về địa hình du lịch mong muốn tới

+ Thông tin về số lượng người tham gia du lịch

+ Thông tin về thời gian

+ Thông tin về mục đích du lịch

+ Địa điểm du lịch và đánh giá về địa điểm du lịch đó

- Sau đó thu được hơn 20000 bản ghi được tổng hợp và gửi về.



Ngoài ra, đầu vào mới và kết quả dự đoán sẽ được thêm vào tập dữ liệu để làm cho tập dữ liệu được mạnh mẽ và chính xác hơn.

### 2.3.2. Xử lí dữ liệu

Thu thập dữ liệu là một nhiệm vụ quan trọng và làm cho dữ liệu đó trở nên hữu ích là một nhiệm vụ quan trọng khác.

Dữ liệu được thu thập từ các phương tiện khác nhau sẽ được định dạng không được tổ chức và có thể có nhiều giá trị rỗng, trong giá trị dữ liệu hợp lệ và dữ liệu không mong muốn. Làm sạch tất cả những dữ liệu này, thay thế chúng bằng dữ liệu thích hợp hoặc gần đúng, xóa dữ liệu trống hay thay thế chúng bằng một số giá trị thay thế cố định là các bước cơ bản trong xử lý trước dữ liệu.

Ngay cả dữ liệu được thu thập cũng có thể chứa giá trị rác. Nó có thể không có định dạng hoặc cách thức chính xác. Tất cả trường hợp như vậy phải được xác minh và thay thế bằng các giá trị thay thế để làm cho dữ liệu trở nên có ý nghĩa và hữu ích hơn nữa.

Đây là yếu tố tiên quyết quan trọng cho việc huấn luyện mô hình nên cần rất nghiêm túc cho việc xử lí dữ liệu này.

Quá trình xử lí gồm các bước như sau:

\* Lọc bỏ dữ liệu rác (thiếu thông tin, trùng lặp, giá trị thông tin không chính xác,…)

Sau quá trình xử lí, tập dữ liệu của tôi gồm 14505 bản ghi được thu thập và xử lí kĩ càng, được định dạng dưới dạng file exel.

Gồm 14 cột bao gồm:

- Địa hình: Thành phố, Núi, Biển

- Mức thu nhập: Dưới 5 triệu, 5-15 triệu, 15-30 triệu, Trên 30 triệu

- Số lượng người: Một mình, Cặp đôi, 3-5 người, 6-10 người, Trên 10 người.

- Thời gian: Xuân, Hè, Thu, Đông.

- Mục đích du lịch: Nghỉ dưỡng, Trải nghiệm, Tìm hiểu văn hóa, Tìm hiểu lịch sử, Tâm linh, Khám phá thiên nhiên.

- Địa điểm: 20 địa điểm du lịch tại Huế: Đồi Thiên An – Hồ Thuỷ Tiên, Núi Bạch Mã, Chùa Huyền Không Sơn Thượng, Biển Lăng Cô, Biển Thuận An, Khu nghỉ dưỡng Banyan Tree, Khu nghỉ dưỡng Pilgrimage Village, Chùa Thiên Mụ, Chùa Từ Đàm, Bảo tàng Mỹ thuật Cung đình Huế, Thiền viện Trúc Lâm Bạch Mã, Nhà vườn Huế, Lăng tẩm Huế, Cầu Tràng Tiền, Đại Nội Huế, Sông Hương, Phá Tam Giang, Phố đi bộ Huế, Chợ Đông Ba, Đèo Hải Vân.

- Đánh giá: 6 nhãn từ 0 -> 5\*.



\* Trực quan hóa dữ liệu để quan sát, đánh giá tính chất tập dữ liệu.

Quan sát dữ liệu theo các trường thông tin, đặc biệt là biến mục tiêu “Đánh giá”. Như quan sát ở hình dưới, dữ liệu bị mất cân bằng khá lớn.

- Mức đánh giá 0\*: 1256 mẫu

- Mức đánh giá 1\*: 3921 mẫu

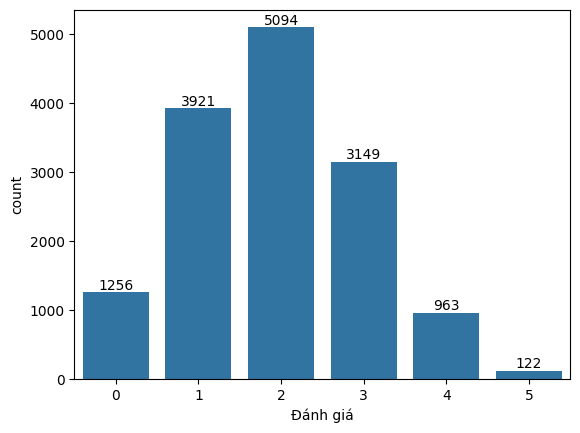
- Mức đánh giá 2\*: 5094 mẫu

- Mức đánh giá 4\*: 3149 mẫu

- Mức đánh giá 4\*: 963 mẫu

- Mức đánh giá 5\*: 122 mẫu

Vì vậy cần xem xét áp dụng các kĩ thuật cân bằng dữ liệu cho tập dữ liệu huấn luyện.



Hình 2. : Trực quan hóa dữ liệu về Mức đánh giá

### 2.3.3. Phân chia tập dữ liệu

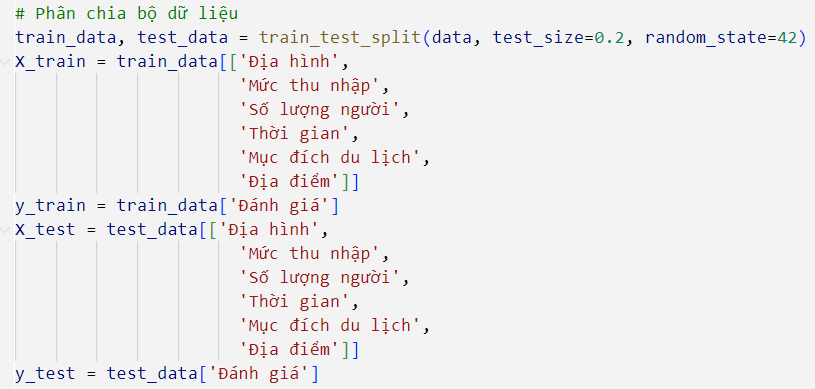
Với tập dữ liệu trên, phương pháp đánh giá train – test dataset được sử dụng để đánh giá mô hình. Tập dữ liệu được chia thành 2 bộ gồm 80% cho bộ train và 20% cho bộ test.

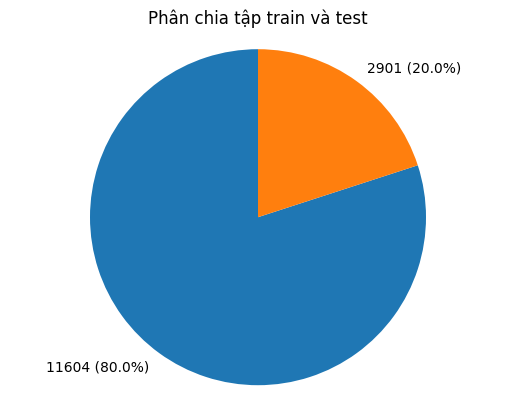
Sử dụng phương thức *train\_test\_split* từ module *sklearn.model\_selection* để phân chia tập dữ liệu với tham số đầu vào:

- *data*: bộ dữ liệu

- *test\_size*: Độ lớn bộ dữ liệu test. Ở đây lấy bằng 0.2 tức là bộ test chiếm 20% tổng dữ liệu.

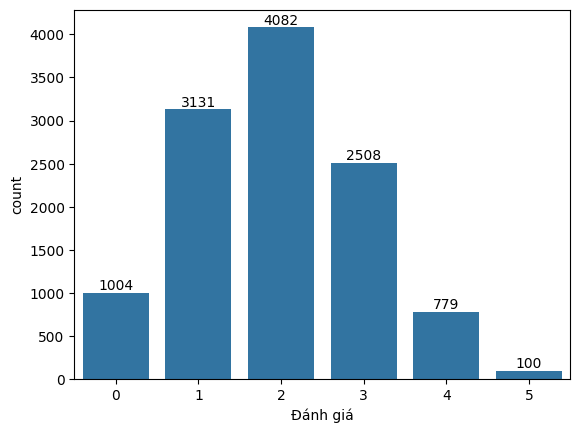
- *random*\_*state*: Hệ số ngẫu nhiên





Hình 2. : Phân chia tập dữ liệu

Sau khi phân chi được tập dữ liệu huấn luyện như sau:



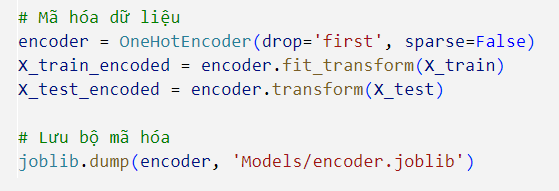
Hình 2. : Tập dữ liệu huấn luyện theo Mức đánh giá

### 2.3.4. Mã hóa dữ liệu

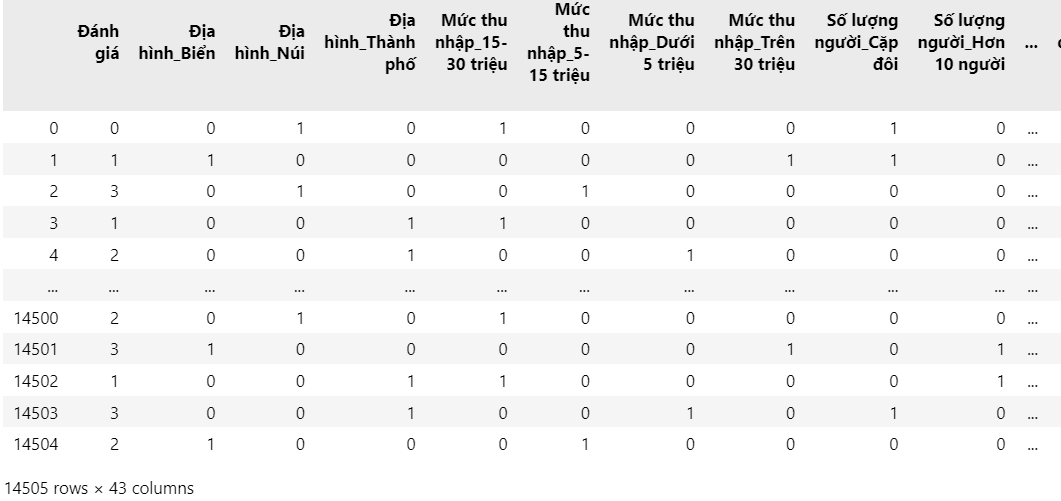
Dữ liệu để đưa vào huấn luyện mô hình học máy cần được mã hóa cho phù hợp. Đối với dữ liệu dạng nhãn như trên, áp dụng cho bài toán phân lớp, dữ liệu được mã hóa kiểu One-hot, tức là mỗi ô dữ liệu sẽ được mã hóa dạng nhị phân của trường thông tin đó.

Ví dụ: Dữ liệu “Dưới 5 triệu” trong trường thông tin Mức thu nhập (Dưới 5 triệu, 5-15 triệu, 15-30 triệu, Trên 30 triệu) được mã hóa là [1, 0, 0, 0]

Sử dụng phương thức *OneHotEncoder()* từ thư viện *sklearn.preprocessing import* để tạo bộ mã hóa One-hot.



Sau khi mã hóa, thu được bộ dữ liệu như sau:



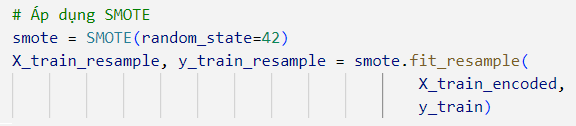
Hình 2. : Dữ liệu sau khi mã hóa One-hot

Bộ mã hóa được lưu vào file *encoder.joblib* trong thư mục Models của thư mục dự án.

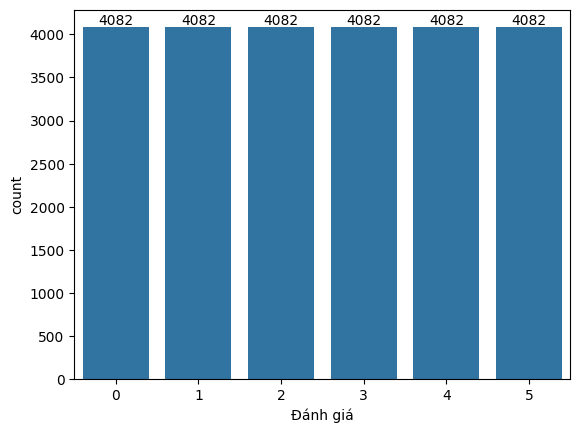
### 2.3.5. Cân bằng dữ liệu

Đánh giá tập dữ liệu bị mất cân bằng khá lớn, tức là các nhãn mục tiêu chênh lệch nhau lớn về số lượng mẫu. Thực hiện phương án cân bằng dữ liệu SMOTE để cân bằng dữ liệu tối ưu cho mô hình học máy.

Sử dụng phương thức SMOTE() lớp *imblearn.over\_sampling* với tập dữ liệu huấn luyện để cân bằng dữ liệu.



Sau khi xử lí, ta thấy dữ liệu đã được thêm mẫu và cân bằng



Hình 2. : Trực quan dữ liệu theo Mức đánh giá sau khi cân bằng

## 2.4. Hiệu chỉnh tham số mô hình học máy

Mỗi mô hình học máy đều có những ưu nhược điểm riêng, việc xem xét sử dụng mô hình nào thường được thực hiện một các định lượng bằng các thử nghiệm với nhiều mô hình và chọn ra mô hình phù hợp nhất với dữ liệu và bảo đảm được tính hiệu quả cần thiết. Trong đề tài sử dụng 4 mô hình học máy phổ biến và mạnh mẽ nhất hiện này phù hợp với kiểu dữ liệu đã thu thập để để đánh giá đó là Cây quyết định, Rừng cây ngẫu nhiên, Xgboost, SVM.

Để huấn luyện mô hình hiệu quả nhất, cần hiệu chỉnh và chọn các tham số cho mô hình được tối ưu nhất. Ta sử dụng xác thực chéo để hiệu chỉnh mô hình học máy trên lưới tham số. Cụ thể như sau:

### 2.4.1. Chọn tham số cho mô hình Cây quyết định

Sử dụng phương thức *GridSearchCV()* để từ thư viện *sklearn.model\_selection* để thực hiện xác thực chéo các bộ tham số, với ma trận tham số:

*{*

*'criterion': ['gini', 'entropy'],*

*'max\_depth': [None, 5, 10, 15],*

*'min\_samples\_split': [2, 5, 10],*

*'min\_samples\_leaf': [1, 2, 4],*

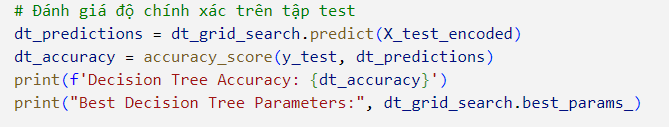
*}*



Sau khi xác thực, mô hình xác thực được lưu vào file *grid\_search\_dt\_model.joblib* trong thư mục Models của thư mục dự án. Còn kết quả xác thực được lưu vào fiel *grid\_search\_dt\_result.xlsx* trong thư mục Results. Thực hiện xác thực thu được kết quả như sau:

Bảng 2. : Bảng kết quả hiện chỉnh tham số mô hình Cây quyết định

|  |  |  |
| --- | --- | --- |
| **params** | **mean\_test\_score** | **rank\_test\_score** |
| {'criterion': 'gini', 'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} | 0.850904 | 1 |
| {'criterion': 'entropy', 'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} | 0.848111 | 2 |
| {'criterion': 'gini', 'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 5} | 0.835706 | 3 |
| {'criterion': 'entropy', 'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 5} | 0.833548 | 4 |
| {'criterion': 'entropy', 'max\_depth': None, 'min\_samples\_leaf': 2, 'min\_samples\_split': 2} | 0.824277 | 5 |
| {'criterion': 'gini', 'max\_depth': None, 'min\_samples\_leaf': 2, 'min\_samples\_split': 2} | 0.821059 | 6 |
| {'criterion': 'entropy', 'max\_depth': None, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5} | 0.82072 | 7 |
| {'criterion': 'gini', 'max\_depth': None, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5} | 0.81945 | 8 |
| {'criterion': 'gini', 'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10} | 0.807131 | 9 |
| {'criterion': 'entropy', 'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10} | 0.805396 | 10 |
| {'criterion': 'gini', 'max\_depth': None, 'min\_samples\_leaf': 2, 'min\_samples\_split': 10} | 0.800569 | 11 |
| {'criterion': 'entropy', 'max\_depth': None, 'min\_samples\_leaf': 2, 'min\_samples\_split': 10} | 0.800443 | 12 |
| {'criterion': 'gini', 'max\_depth': None, 'min\_samples\_leaf': 4, 'min\_samples\_split': 2} | 0.783552 | 13 |
| {'criterion': 'gini', 'max\_depth': None, 'min\_samples\_leaf': 4, 'min\_samples\_split': 5} | 0.783552 | 13 |
| {'criterion': 'entropy', 'max\_depth': None, 'min\_samples\_leaf': 4, 'min\_samples\_split': 2} | 0.782875 | 15 |
| {'criterion': 'entropy', 'max\_depth': None, 'min\_samples\_leaf': 4, 'min\_samples\_split': 5} | 0.782875 | 15 |
| {'criterion': 'gini', 'max\_depth': None, 'min\_samples\_leaf': 4, 'min\_samples\_split': 10} | 0.779911 | 17 |
| {'criterion': 'entropy', 'max\_depth': None, 'min\_samples\_leaf': 4, 'min\_samples\_split': 10} | 0.779192 | 18 |
| {'criterion': 'entropy', 'max\_depth': 15, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} | 0.696556 | 19 |
| {'criterion': 'entropy', 'max\_depth': 15, 'min\_samples\_leaf': 1, 'min\_samples\_split': 5} | 0.694313 | 20 |
| {'criterion': 'gini', 'max\_depth': 15, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} | 0.691434 | 21 |
| {'criterion': 'entropy', 'max\_depth': 15, 'min\_samples\_leaf': 2, 'min\_samples\_split': 2} | 0.691307 | 22 |
| {'criterion': 'entropy', 'max\_depth': 15, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5} | 0.690799 | 23 |
| {'criterion': 'gini', 'max\_depth': 15, 'min\_samples\_leaf': 1, 'min\_samples\_split': 5} | 0.689953 | 24 |
| {'criterion': 'entropy', 'max\_depth': 15, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10} | 0.688851 | 25 |
| {'criterion': 'entropy', 'max\_depth': 15, 'min\_samples\_leaf': 2, 'min\_samples\_split': 10} | 0.686523 | 26 |
| {'criterion': 'gini', 'max\_depth': 15, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5} | 0.684788 | 27 |
| {'criterion': 'gini', 'max\_depth': 15, 'min\_samples\_leaf': 2, 'min\_samples\_split': 2} | 0.684746 | 28 |
| {'criterion': 'gini', 'max\_depth': 15, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10} | 0.683306 | 29 |
| {'criterion': 'gini', 'max\_depth': 15, 'min\_samples\_leaf': 2, 'min\_samples\_split': 10} | 0.681867 | 30 |
| {'criterion': 'entropy', 'max\_depth': 15, 'min\_samples\_leaf': 4, 'min\_samples\_split': 2} | 0.680808 | 31 |
| {'criterion': 'entropy', 'max\_depth': 15, 'min\_samples\_leaf': 4, 'min\_samples\_split': 5} | 0.680808 | 31 |
| {'criterion': 'entropy', 'max\_depth': 15, 'min\_samples\_leaf': 4, 'min\_samples\_split': 10} | 0.680385 | 33 |
| {'criterion': 'gini', 'max\_depth': 15, 'min\_samples\_leaf': 4, 'min\_samples\_split': 2} | 0.675051 | 34 |
| {'criterion': 'gini', 'max\_depth': 15, 'min\_samples\_leaf': 4, 'min\_samples\_split': 5} | 0.675051 | 34 |
| {'criterion': 'gini', 'max\_depth': 15, 'min\_samples\_leaf': 4, 'min\_samples\_split': 10} | 0.674501 | 36 |
| {'criterion': 'entropy', 'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} | 0.575864 | 37 |
| {'criterion': 'entropy', 'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 5} | 0.575356 | 38 |
| {'criterion': 'entropy', 'max\_depth': 10, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5} | 0.574805 | 39 |
| {'criterion': 'entropy', 'max\_depth': 10, 'min\_samples\_leaf': 2, 'min\_samples\_split': 2} | 0.574636 | 40 |
| {'criterion': 'entropy', 'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10} | 0.574339 | 41 |
| {'criterion': 'entropy', 'max\_depth': 10, 'min\_samples\_leaf': 2, 'min\_samples\_split': 10} | 0.574128 | 42 |
| {'criterion': 'gini', 'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} | 0.573451 | 43 |
| {'criterion': 'gini', 'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 5} | 0.573112 | 44 |
| {'criterion': 'gini', 'max\_depth': 10, 'min\_samples\_leaf': 2, 'min\_samples\_split': 2} | 0.572053 | 45 |
| {'criterion': 'gini', 'max\_depth': 10, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5} | 0.571884 | 46 |
| {'criterion': 'gini', 'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10} | 0.571884 | 47 |
| {'criterion': 'entropy', 'max\_depth': 10, 'min\_samples\_leaf': 4, 'min\_samples\_split': 2} | 0.571588 | 48 |
| {'criterion': 'entropy', 'max\_depth': 10, 'min\_samples\_leaf': 4, 'min\_samples\_split': 5} | 0.571588 | 48 |
| {'criterion': 'entropy', 'max\_depth': 10, 'min\_samples\_leaf': 4, 'min\_samples\_split': 10} | 0.571546 | 50 |
| {'criterion': 'gini', 'max\_depth': 10, 'min\_samples\_leaf': 2, 'min\_samples\_split': 10} | 0.570656 | 51 |
| {'criterion': 'gini', 'max\_depth': 10, 'min\_samples\_leaf': 4, 'min\_samples\_split': 2} | 0.569132 | 52 |
| {'criterion': 'gini', 'max\_depth': 10, 'min\_samples\_leaf': 4, 'min\_samples\_split': 5} | 0.569132 | 52 |
| {'criterion': 'gini', 'max\_depth': 10, 'min\_samples\_leaf': 4, 'min\_samples\_split': 10} | 0.569048 | 54 |
| {'criterion': 'gini', 'max\_depth': 5, 'min\_samples\_leaf': 4, 'min\_samples\_split': 2} | 0.386378 | 55 |
| {'criterion': 'gini', 'max\_depth': 5, 'min\_samples\_leaf': 4, 'min\_samples\_split': 5} | 0.386378 | 55 |
| {'criterion': 'gini', 'max\_depth': 5, 'min\_samples\_leaf': 4, 'min\_samples\_split': 10} | 0.386378 | 55 |
| {'criterion': 'gini', 'max\_depth': 5, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} | 0.386336 | 58 |
| {'criterion': 'gini', 'max\_depth': 5, 'min\_samples\_leaf': 1, 'min\_samples\_split': 5} | 0.386336 | 58 |
| {'criterion': 'gini', 'max\_depth': 5, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10} | 0.386336 | 58 |
| {'criterion': 'gini', 'max\_depth': 5, 'min\_samples\_leaf': 2, 'min\_samples\_split': 2} | 0.386336 | 58 |
| {'criterion': 'gini', 'max\_depth': 5, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5} | 0.386336 | 58 |
| {'criterion': 'gini', 'max\_depth': 5, 'min\_samples\_leaf': 2, 'min\_samples\_split': 10} | 0.386336 | 58 |
| {'criterion': 'entropy', 'max\_depth': 5, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} | 0.384092 | 64 |
| {'criterion': 'entropy', 'max\_depth': 5, 'min\_samples\_leaf': 1, 'min\_samples\_split': 5} | 0.384092 | 64 |
| {'criterion': 'entropy', 'max\_depth': 5, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10} | 0.384092 | 64 |
| {'criterion': 'entropy', 'max\_depth': 5, 'min\_samples\_leaf': 2, 'min\_samples\_split': 2} | 0.384092 | 64 |
| {'criterion': 'entropy', 'max\_depth': 5, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5} | 0.384092 | 64 |
| {'criterion': 'entropy', 'max\_depth': 5, 'min\_samples\_leaf': 2, 'min\_samples\_split': 10} | 0.384092 | 64 |
| {'criterion': 'entropy', 'max\_depth': 5, 'min\_samples\_leaf': 4, 'min\_samples\_split': 2} | 0.38405 | 70 |
| {'criterion': 'entropy', 'max\_depth': 5, 'min\_samples\_leaf': 4, 'min\_samples\_split': 5} | 0.38405 | 70 |
| {'criterion': 'entropy', 'max\_depth': 5, 'min\_samples\_leaf': 4, 'min\_samples\_split': 10} | 0.38405 | 70 |



Đánh giá độ chính xác trên tập test thu được kết quả độ chính xác:

*Decision Tree Accuracy: 0.7736696613683484*

Sau khi xác thực, ta thu được mô hình tốt nhất với tham số;

*Best Decision Tree Parameters: {'criterion': 'gini', 'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2}*

### 2.4.2. Chọn tham số cho mô hình Rừng cây ngẫu nhiên

Sử dụng phương thức *GridSearchCV()* để từ thư viện *sklearn.model\_selection* để thực hiện xác thực chéo các bộ tham số, với ma trận tham số:

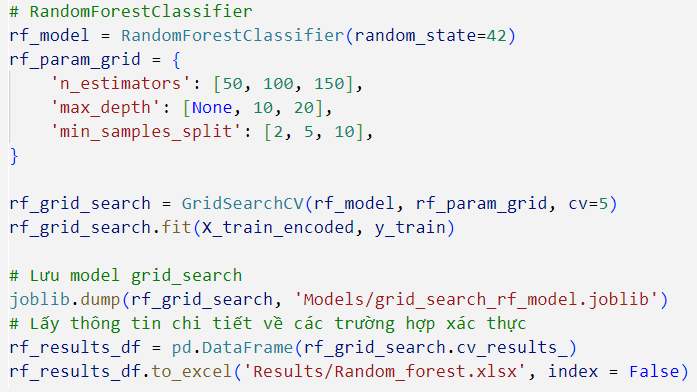
*{*

*'n\_estimators': [50, 100, 150],*

*'max\_depth': [None, 10, 20],*

*'min\_samples\_split': [2, 5, 10],*

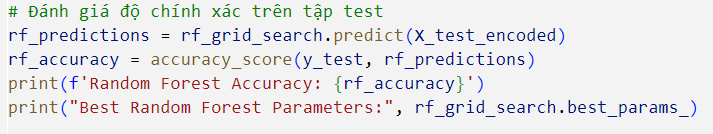
*}*



Sau khi xác thực, mô hình xác thực được lưu vào file *grid\_search\_rf\_model.joblib* trong thư mục Models của thư mục dự án. Còn kết quả xác thực được lưu vào fiel *grid\_search\_rft\_result.xlsx* trong thư mục Results. Thực hiện xác thực thu được kết quả như sau:

Bảng 2. : Bảng kết quả hiện chỉnh tham số mô hình Rừng cây ngẫu nhiên

|  |  |  |
| --- | --- | --- |
| **params** | **mean\_test\_score** | **rank\_test\_score** |
| {'max\_depth': None, 'min\_samples\_split': 2, 'n\_estimators': 150} | 0.855519 | 1 |
| {'max\_depth': None, 'min\_samples\_split': 2, 'n\_estimators': 100} | 0.855096 | 2 |
| {'max\_depth': None, 'min\_samples\_split': 5, 'n\_estimators': 150} | 0.854249 | 3 |
| {'max\_depth': None, 'min\_samples\_split': 5, 'n\_estimators': 100} | 0.853783 | 4 |
| {'max\_depth': 20, 'min\_samples\_split': 2, 'n\_estimators': 150} | 0.850735 | 5 |
| {'max\_depth': None, 'min\_samples\_split': 2, 'n\_estimators': 50} | 0.849169 | 6 |
| {'max\_depth': None, 'min\_samples\_split': 5, 'n\_estimators': 50} | 0.847475 | 7 |
| {'max\_depth': 20, 'min\_samples\_split': 2, 'n\_estimators': 100} | 0.846883 | 8 |
| {'max\_depth': 20, 'min\_samples\_split': 2, 'n\_estimators': 50} | 0.840321 | 9 |
| {'max\_depth': 20, 'min\_samples\_split': 5, 'n\_estimators': 150} | 0.839347 | 10 |
| {'max\_depth': 20, 'min\_samples\_split': 5, 'n\_estimators': 100} | 0.836511 | 11 |
| {'max\_depth': None, 'min\_samples\_split': 10, 'n\_estimators': 150} | 0.835114 | 12 |
| {'max\_depth': None, 'min\_samples\_split': 10, 'n\_estimators': 100} | 0.832701 | 13 |
| {'max\_depth': 20, 'min\_samples\_split': 5, 'n\_estimators': 50} | 0.832489 | 14 |
| {'max\_depth': None, 'min\_samples\_split': 10, 'n\_estimators': 50} | 0.827155 | 15 |
| {'max\_depth': 20, 'min\_samples\_split': 10, 'n\_estimators': 150} | 0.823133 | 16 |
| {'max\_depth': 20, 'min\_samples\_split': 10, 'n\_estimators': 100} | 0.818688 | 17 |
| {'max\_depth': 20, 'min\_samples\_split': 10, 'n\_estimators': 50} | 0.815259 | 18 |
| {'max\_depth': 10, 'min\_samples\_split': 2, 'n\_estimators': 150} | 0.719586 | 19 |
| {'max\_depth': 10, 'min\_samples\_split': 2, 'n\_estimators': 100} | 0.719459 | 20 |
| {'max\_depth': 10, 'min\_samples\_split': 5, 'n\_estimators': 150} | 0.714887 | 21 |
| {'max\_depth': 10, 'min\_samples\_split': 5, 'n\_estimators': 100} | 0.71476 | 22 |
| {'max\_depth': 10, 'min\_samples\_split': 10, 'n\_estimators': 150} | 0.710611 | 23 |
| {'max\_depth': 10, 'min\_samples\_split': 10, 'n\_estimators': 100} | 0.707479 | 24 |
| {'max\_depth': 10, 'min\_samples\_split': 2, 'n\_estimators': 50} | 0.706039 | 25 |
| {'max\_depth': 10, 'min\_samples\_split': 5, 'n\_estimators': 50} | 0.703499 | 26 |
| {'max\_depth': 10, 'min\_samples\_split': 10, 'n\_estimators': 50} | 0.695202 | 27 |



Đánh giá độ chính xác trên tập test thu được kết quả độ chính xác:

*Random Forest Accuracy: 0.6931582584657913*

Sau khi xác thực, ta thu được mô hình tốt nhất với tham số;

*Best Random Forest Parameters: {'max\_depth': None, 'min\_samples\_split': 2, 'n\_estimators': 150}*

### 2.4.3. Chọn tham số cho mô hình Xgboost

Sử dụng phương thức *GridSearchCV()* để từ thư viện *sklearn.model\_selection* để thực hiện xác thực chéo các bộ tham số, với ma trận tham số:

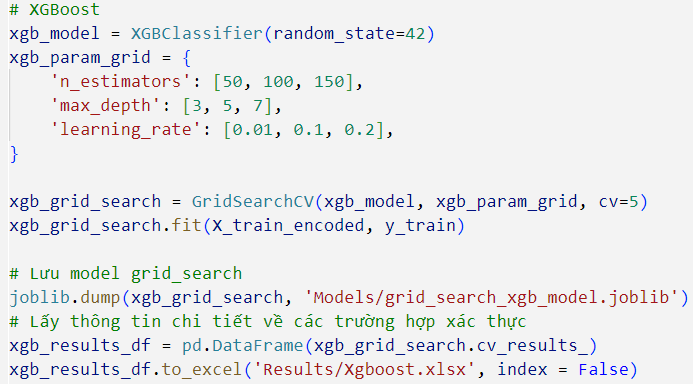
*{*

*'n\_estimators': [50, 100, 150],*

*'max\_depth': [3, 5, 7],*

*'learning\_rate': [0.01, 0.1, 0.2],*

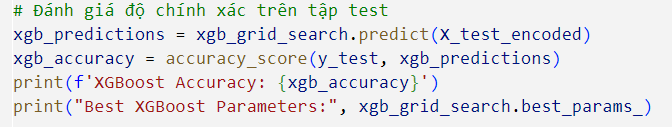
*}*



Sau khi xác thực, mô hình xác thực được lưu vào file *grid\_search\_xgb\_model.joblib* trong thư mục Models của thư mục dự án. Còn kết quả xác thực được lưu vào fiel *grid\_search\_xgb\_result.xlsx* trong thư mục Results. Thực hiện xác thực thu được kết quả như sau:

Bảng 2. 3: Bảng kết quả hiện chỉnh tham số mô hình XGBoost

|  |  |  |
| --- | --- | --- |
| **params** | **score** | **rank** |
| {'learning\_rate': 0.2, 'max\_depth': 7, 'n\_estimators': 150} | 0.868219 | 1 |
| {'learning\_rate': 0.2, 'max\_depth': 7, 'n\_estimators': 100} | 0.83359 | 2 |
| {'learning\_rate': 0.1, 'max\_depth': 7, 'n\_estimators': 150} | 0.810265 | 3 |
| {'learning\_rate': 0.2, 'max\_depth': 5, 'n\_estimators': 150} | 0.775044 | 4 |
| {'learning\_rate': 0.2, 'max\_depth': 7, 'n\_estimators': 50} | 0.774028 | 5 |
| {'learning\_rate': 0.1, 'max\_depth': 7, 'n\_estimators': 100} | 0.771953 | 6 |
| {'learning\_rate': 0.2, 'max\_depth': 5, 'n\_estimators': 100} | 0.738765 | 7 |
| {'learning\_rate': 0.1, 'max\_depth': 5, 'n\_estimators': 150} | 0.713703 | 8 |
| {'learning\_rate': 0.1, 'max\_depth': 7, 'n\_estimators': 50} | 0.703246 | 9 |
| {'learning\_rate': 0.1, 'max\_depth': 5, 'n\_estimators': 100} | 0.674841 | 10 |
| {'learning\_rate': 0.2, 'max\_depth': 5, 'n\_estimators': 50} | 0.672342 | 11 |
| {'learning\_rate': 0.2, 'max\_depth': 3, 'n\_estimators': 150} | 0.641186 | 12 |
| {'learning\_rate': 0.1, 'max\_depth': 5, 'n\_estimators': 50} | 0.607487 | 13 |
| {'learning\_rate': 0.2, 'max\_depth': 3, 'n\_estimators': 100} | 0.60698 | 14 |
| {'learning\_rate': 0.01, 'max\_depth': 7, 'n\_estimators': 150} | 0.605327 | 15 |
| {'learning\_rate': 0.01, 'max\_depth': 7, 'n\_estimators': 100} | 0.585557 | 16 |
| {'learning\_rate': 0.1, 'max\_depth': 3, 'n\_estimators': 150} | 0.585178 | 17 |
| {'learning\_rate': 0.01, 'max\_depth': 7, 'n\_estimators': 50} | 0.569512 | 18 |
| {'learning\_rate': 0.2, 'max\_depth': 3, 'n\_estimators': 50} | 0.551987 | 19 |
| {'learning\_rate': 0.1, 'max\_depth': 3, 'n\_estimators': 100} | 0.549574 | 20 |
| {'learning\_rate': 0.01, 'max\_depth': 5, 'n\_estimators': 150} | 0.51431 | 21 |
| {'learning\_rate': 0.01, 'max\_depth': 5, 'n\_estimators': 100} | 0.499578 | 22 |
| {'learning\_rate': 0.1, 'max\_depth': 3, 'n\_estimators': 50} | 0.488021 | 23 |
| {'learning\_rate': 0.01, 'max\_depth': 5, 'n\_estimators': 50} | 0.487047 | 24 |
| {'learning\_rate': 0.01, 'max\_depth': 3, 'n\_estimators': 150} | 0.417535 | 25 |
| {'learning\_rate': 0.01, 'max\_depth': 3, 'n\_estimators': 100} | 0.402719 | 26 |
| {'learning\_rate': 0.01, 'max\_depth': 3, 'n\_estimators': 50} | 0.392347 | 27 |



Đánh giá độ chính xác trên tập test thu được kết quả độ chính xác:

*XGBoost Accuracy: 0.7964754664823773*

Sau khi xác thực, ta thu được mô hình tốt nhất với tham số;

*Best XGBoost Parameters: {'learning\_rate': 0.2, 'max\_depth': 7, 'n\_estimators': 150}*

### 2.4.4. Chọn tham số cho mô hình Máy vector hỗ trợ

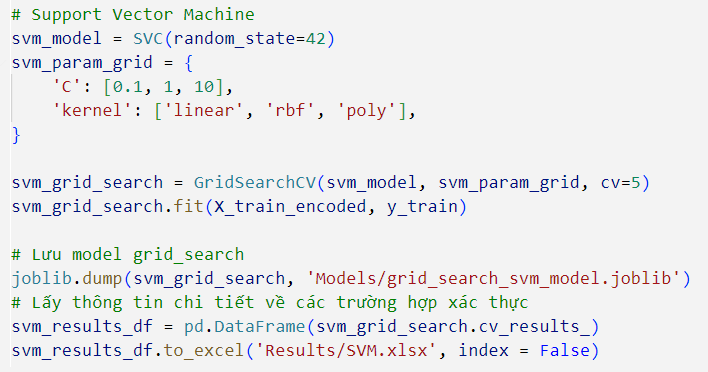
Sử dụng phương thức *GridSearchCV()* để từ thư viện *sklearn.model\_selection* để thực hiện xác thực chéo các bộ tham số, với ma trận tham số:

*{*

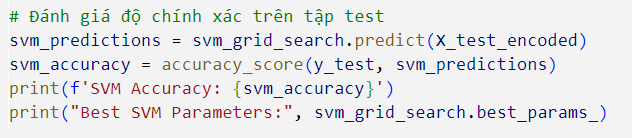
*'C': [0.1, 1, 10],*

*'kernel': ['linear', 'rbf', 'poly'],*

*}*



Sau khi xác thực, mô hình xác thực được lưu vào file *grid\_search\_svm\_model.joblib* trong thư mục Models của thư mục dự án. Còn kết quả xác thực được lưu vào fiel *grid\_search\_svm\_result.xlsx* trong thư mục Results. Thực hiện xác thực thu được kết quả như sau:**Error! Not a valid link.**



Đánh giá độ chính xác trên tập test thu được kết quả độ chính xác:

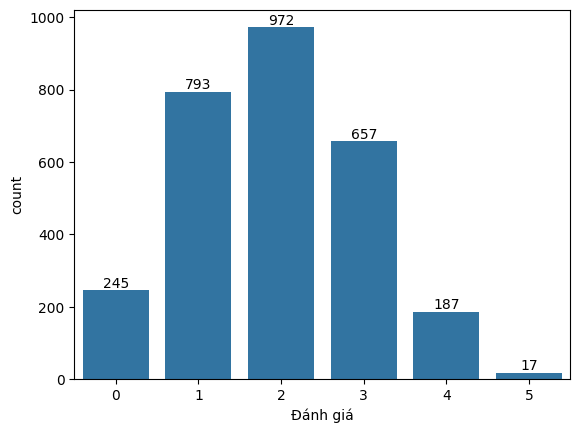
*SVM Accuracy: 0.9882515549412578*

Sau khi xác thực, ta thu được mô hình tốt nhất với tham số:

*Best SVM Parameters: {'C': 10, 'kernel': 'rbf'}*

## 2.5. Huấn luyện và đánh giá mô hình

Để đánh giá mô hình sử dụng tập dữ liệu test (20% của bộ dữ liệu) như đã phân chia. Sử dụng các tham số đã chọn ở trên để huấn luyện các mô hình với các thuật toán Decision Tree, Random Forest, XGBoost, Support Vector Machine.



Hình 2. : Trực quan hóa dữ liệu test theo Mức đánh giá

Thực hiện đánh giá dựa trên các chỉ số về Độ chính xác (Accuracy),Precision, Recall, F1-score. Tiến hành huấn luyện và đánh giá các mô hình với tham số tối ưu như đã hiệu chỉnh được ở trên.

### 2.5.1. Đánh giá mô hình Cây quyết định

Sử dụng lớp *DecisionTreeClassifier*: Là một mô hình cây quyết định trong thư viện scikit-learn, để thực hiện huấn luyện mô hình. Trong đó các tham số truyền vào:

*- random\_state*=42: Đảm bảo kết quả của mô hình có thể được tái tạo bằng cách sử dụng một số nguyên cụ thể.

*criterion*='gini': Sử dụng chỉ số Gini để đo lường sự không thuần khiết của các nhóm.

*max\_depth*=None: Không giới hạn độ sâu của cây.

*min\_samples\_leaf*=1: Số mẫu tối thiểu cần có trong mỗi lá của cây.

*min\_samples\_split*=2: Số mẫu tối thiểu cần để một nút có thể được chia.

Thời gian thực hiện việc huấn luyện mô hình được đo lường từ *start\_time* đến *end\_time*.

*encoder.transform*(*X\_test*): Mã hóa dữ liệu kiểm thử bằng encoder (trước đó đã được sử dụng để mã hóa dữ liệu huấn luyện).

*accuracy\_score*: Đo lường độ chính xác của mô hình bằng cách so sánh dự đoán với nhãn thực tế trong dữ liệu kiểm thử.

Lưu mô hình đã huấn luyện vào một tệp *best\_dt\_model.joblib* trong thư mục Models của thư mục dự án.

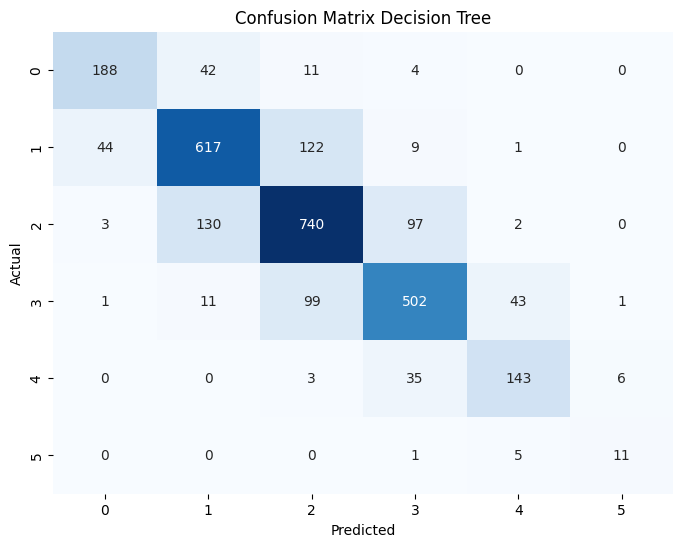


Tính toán và in ra báo cáo phân loại, sau đó lưu báo cáo vào một tệp tin *report\_dt.xlsx* trong thư mục Results của thư mục dự án.

Bảng 2. 4: Kết quả đánh giá mô hình Decision Tree trên tập dữ liệu kiểm thử

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decision Tree** | **precision** | **recall** | **f1-score** | **support** |
| **0** | 0.8 | 0.77 | 0.78 | 245 |
| **1** | 0.77 | 0.78 | 0.77 | 793 |
| **2** | 0.76 | 0.76 | 0.76 | 972 |
| **3** | 0.77 | 0.76 | 0.77 | 657 |
| **4** | 0.74 | 0.76 | 0.75 | 187 |
| **5** | 0.61 | 0.65 | 0.63 | 17 |
| **accuracy** |  |  | 0.77 | 2871 |
| **macro avg** | 0.74 | 0.75 | 0.74 | 2871 |
| **weighted avg** | 0.77 | 0.77 | 0.77 | 2871 |
| **Time** | 0.08833432197570801s | | | |

Tính toán và vẽ ma trận nhầm lẫn để đánh giá hiệu suất của mô hình trên dữ liệu kiểm thử.



Hình 2. : Mã trận lỗi của mô hình Decision Tree

### 2.5.2. Đánh giá mô hình Rừng cây ngẫu nhiên

Sử dụng lớp *RandomForestClassifier*: Là một mô hình Random Forest trong thư viện scikit-learn. Sử dụng tham số như đã hiệu chỉnh ở trên:

*- random\_state*=42: Đảm bảo kết quả có thể tái tạo.

*- max\_depth*=None: Không giới hạn độ sâu của cây trong mỗi "cây" (estimator) của Random Forest.

*- min\_samples\_split*=2: Số mẫu tối thiểu cần để một nút có thể được chia.

*- n\_estimators*=150: Số cây quyết định (estimators) trong Random Forest.

Thời gian thực hiện việc huấn luyện mô hình được đo lường từ *start\_time* đến *end\_time*.

*encoder.transform*(*X\_test*): Mã hóa dữ liệu kiểm thử bằng encoder (trước đó đã được sử dụng để mã hóa dữ liệu huấn luyện).

*accuracy\_score*: Đo lường độ chính xác của mô hình bằng cách so sánh dự đoán với nhãn thực tế trong dữ liệu kiểm thử.

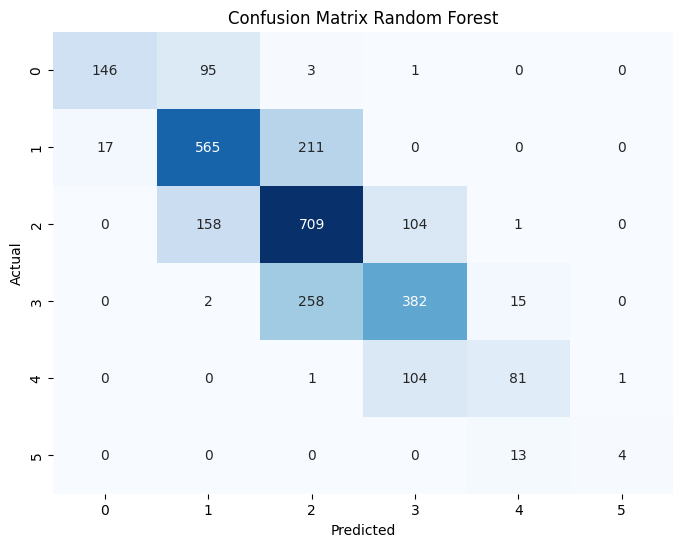
Lưu mô hình đã huấn luyện vào một tệp *best\_rf\_model.joblib* trong thư mục Models của thư mục dự án.



Tính toán và in ra báo cáo phân loại, sau đó lưu báo cáo vào một tệp tin *report\_dt.xlsx* trong thư mục Results của thư mục dự án.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **0** | 0.9 | 0.6 | 0.72 | 245 |
| **1** | 0.69 | 0.71 | 0.7 | 793 |
| **2** | 0.6 | 0.73 | 0.66 | 972 |
| **3** | 0.65 | 0.58 | 0.61 | 657 |
| **4** | 0.74 | 0.43 | 0.55 | 187 |
| **5** | 0.8 | 0.24 | 0.36 | 17 |
| **accuracy** |  |  | 0.66 | 2871 |
| **macro avg** | 0.73 | 0.55 | 0.6 | 2871 |
| **weighted avg** | 0.67 | 0.66 | 0.66 | 2871 |

Tính toán và vẽ ma trận nhầm lẫn để đánh giá hiệu suất của mô hình trên dữ liệu kiểm thử.



Hình 2. : Ma trận lỗi mô hình Rừng cây quyết định

### 2.5.3.. Đánh giá mô hình XGBoost

Sử dụng lớp *XGBClassifier*: Là một mô hình XGBoost trong thư viện XGBoost. Sử dụng bộ tham số đã hiệu chỉnh ở mục trước:

*- random\_state*=42: Đảm bảo kết quả có thể tái tạo.

*- learning\_rate*=0.2: Tỷ lệ học giảm (shrinkage) để kiểm soát quá mức lớn của các cây.

*- max\_depth*=7: Độ sâu tối đa của mỗi cây quyết định.

*- n\_estimators*=150: Số cây quyết định trong ensemble.

Thời gian thực hiện việc huấn luyện mô hình được đo lường từ *start\_time* đến *end\_time*.

*encoder.transform*(*X\_test*): Mã hóa dữ liệu kiểm thử bằng encoder (trước đó đã được sử dụng để mã hóa dữ liệu huấn luyện).

*accuracy\_score*: Đo lường độ chính xác của mô hình bằng cách so sánh dự đoán với nhãn thực tế trong dữ liệu kiểm thử.

Lưu mô hình đã huấn luyện vào một tệp *best\_xgb\_model.joblib* trong thư mục Models của thư mục dự án.

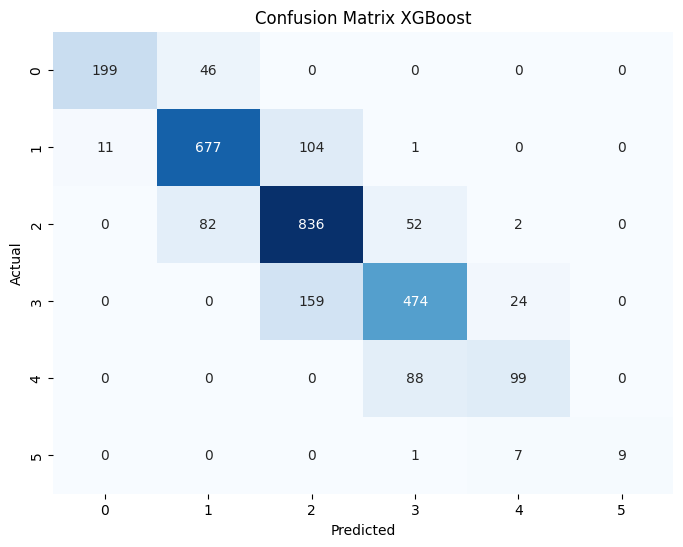


Tính toán và in ra báo cáo phân loại, sau đó lưu báo cáo vào một tệp tin *report\_xgb.xlsx* trong thư mục Results của thư mục dự án.

Bảng 2. 5: Kết quả đánh giá mô hình XBGoost trên tập dữ liệu kiểm thử

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Xgboost | **precision** | **recall** | **f1-score** | **support** |
| **0** | 0.95 | 0.81 | 0.87 | 245 |
| **1** | 0.84 | 0.85 | 0.85 | 793 |
| **2** | 0.76 | 0.86 | 0.81 | 972 |
| **3** | 0.77 | 0.72 | 0.74 | 657 |
| **4** | 0.75 | 0.53 | 0.62 | 187 |
| **5** | 1 | 0.53 | 0.69 | 17 |
| **accuracy** |  |  | 0.8 | 2871 |
| **macro avg** | 0.84 | 0.72 | 0.76 | 2871 |
| **weighted avg** | 0.8 | 0.8 | 0.8 | 2871 |
| **Time** | 1.317885160446167s | | | |

Tính toán và vẽ ma trận nhầm lẫn để đánh giá hiệu suất của mô hình trên dữ liệu kiểm thử.



Hình 2. 21: Ma trận lỗi mô hình Xgboost trên tập dữ liệu kiểm thử

### 2.5.4. Đánh giá mô hình Máy hỗ trợ vector

Sử dụng lớp *SVC*: Là một mô hình Support Vector Machine trong thư viện scikit-learn.

*- random\_state*=42: Đảm bảo kết quả có thể tái tạo.

*- C=*10: Tham số kiểm soát độ chệch (slack variable) của mô hình SVM.

*- kernel*='rbf': Sử dụng hàm nhân RBF (Radial Basis Function) làm hàm nhân (kernel function).

Thời gian thực hiện việc huấn luyện mô hình được đo lường từ *start\_time* đến *end\_time*.

*encoder.transform*(*X\_test*): Mã hóa dữ liệu kiểm thử bằng encoder (trước đó đã được sử dụng để mã hóa dữ liệu huấn luyện).

*accuracy\_score*: Đo lường độ chính xác của mô hình bằng cách so sánh dự đoán với nhãn thực tế trong dữ liệu kiểm thử.

Lưu mô hình đã huấn luyện vào một tệp *best\_svm\_model.joblib* trong thư mục Models của thư mục dự án.

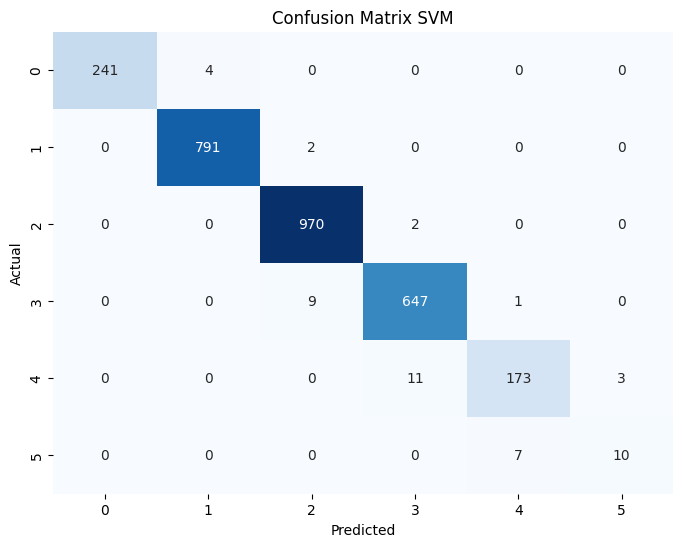


Tính toán và in ra báo cáo phân loại, sau đó lưu báo cáo vào một tệp tin *report\_svm.xlsx* trong thư mục Results của thư mục dự án

Bảng 2. 6: Kết quả đánh giá mô hình XBGoost trên tập dữ liệu kiểm thử

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SVM** | **precision** | **recall** | **f1-score** | **support** |
| **0** | 1 | 0.98 | 0.99 | 245 |
| **1** | 0.99 | 1 | 1 | 793 |
| **2** | 0.99 | 1 | 0.99 | 972 |
| **3** | 0.98 | 0.98 | 0.98 | 657 |
| **4** | 0.96 | 0.93 | 0.94 | 187 |
| **5** | 0.77 | 0.59 | 0.67 | 17 |
| **accuracy** |  |  | 0.99 | 2871 |
| **macro avg** | 0.95 | 0.91 | 0.93 | 2871 |
| **weighted avg** | 0.99 | 0.99 | 0.99 | 2871 |
| **Time** | 22.824801206588745s | | | |

Tính toán và vẽ ma trận nhầm lẫn để đánh giá hiệu suất của mô hình trên dữ liệu kiểm thử.



Hình 2. : Ma trận lỗi mô hình SVM đối với tập dữ liệu kiểm thử

Như vậy, xét về các chỉ số đánh giá, mô hình SVM có kết quả tốt nhất. Như vậy, mô hình SVM với tham số như hiệu chỉnh được lưu lại và đưa vào sử dụng trong chương trình ứng dụng.

## 2.6. Xây dựng giao diện ứng dụng

### 2.6.1. Cấu trúc chương trình

Chương trình được xây dựng theo cấu trúc một dự án ứng dụng web cơ bản. Sử dụng thư viện Flask để tạo dự án. Cấu trúc dự án như sau:

app.py: Chương trình chính của Flask. Định nghĩa một ứng dụng với một route chính ('/') và một hàm index để render template index.html. Định nghĩa một API để trả về thông tin địa điểm du lịch cho người dùng.

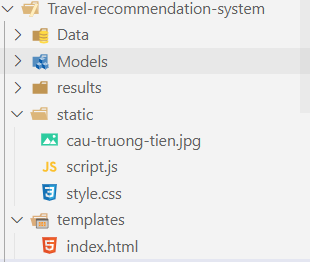
static

|-------script.js: Tệp JavaScript tĩnh.

|-------style.css: Tệp CSS tĩnh.

templates

|-------index.html: Mẫu HTML chính của trang web. Sử dụng {% url\_for('static', filename='...') %} để liên kết đến các tệp tĩnh.



Hình 2. : Cấu trúc dự án Flask

Để chạy ứng dụng, có thể mở terminal, di chuyển đến thư mục chứa *app.py*, và chạy *python app.py*. Sau đó, mở trình duyệt và truy cập http://127.0.0.1:5000/ để xem ứng dụng Flask.

### 2.6.2. Phía Fontend

\* HTML (templates/index.html):

- Biểu Mẫu Người Dùng:

Trang web có một biểu mẫu HTML với các trường nhập khác nhau để người dùng nhập thông tin:

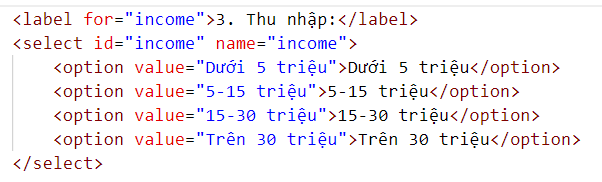
+ Địa hình muốn đến (Núi, Biển, Thành phố).

+ Thời gian muốn đi (Xuân, Hè, Thu, Đông).

+ Thu nhập (Dưới 5 triệu, 5-15 triệu, 15-30 triệu, Trên 30 triệu).

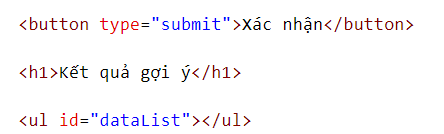
+ Mục đích du lịch (Khám phá thiên nhiên, Tìm hiểu lịch sử, Tìm hiểu văn hóa, Tâm linh, Trải nghiệm, Nghỉ dưỡng).

+ Số người đi (Một mình, Cặp đôi, Nhóm 3-5, Nhóm 6-10, Hơn 10 người).



- Có một nút "Xác nhận" để gửi thông tin từ biểu mẫu lên máy chủ và nhận kết quả dự đoán.

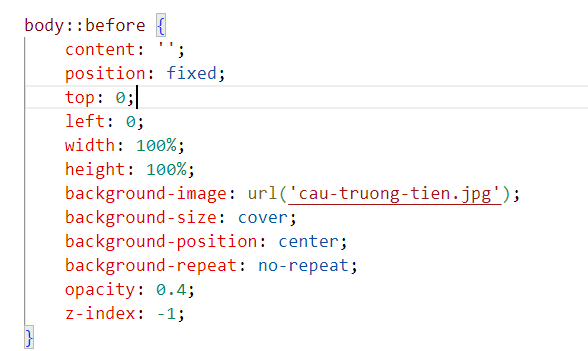
- Dưới biểu mẫu, có một danh sách không thứ tự (<ul>) để hiển thị kết quả gợi ý.



\* CSS (static/style.css): Giao Diện Người Dùng:

- CSS được sử dụng để tạo giao diện người dùng hấp dẫn.

- Nền của trang sử dụng ảnh đẹp và có hiệu ứng mờ.



- Các thành phần như danh sách, nút "Xác nhận" được thiết lập với kiểu dáng đẹp mắt.

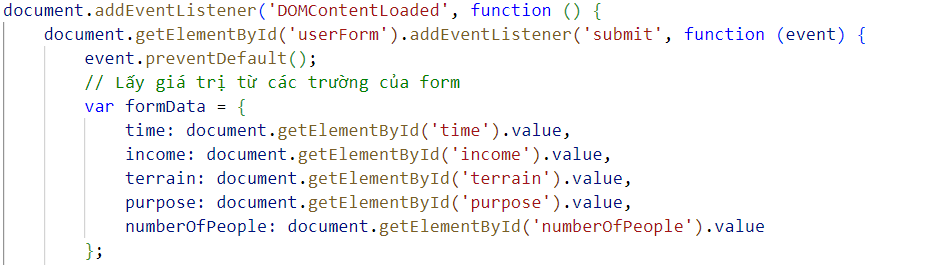
- Các hiệu ứng hover được thêm vào danh sách để cải thiện trải nghiệm người dùng.



\* JavaScript (static/script.js): Tương Tác Người Dùng:

- Sử dụng JavaScript để thêm sự kiện lắng nghe khi người dùng nhấn nút "Xác nhận".

- Xử lý sự kiện submit và ngăn chặn hành vi mặc định của biểu mẫu.

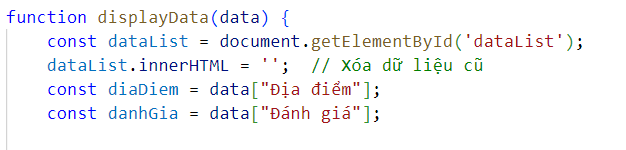


- Sử dụng fetch để gửi yêu cầu POST đến backend và nhận kết quả trả về.

- Hiển thị kết quả trên trang web bằng cách gọi hàm *displayData*(). Cụ thể họa động của hàm như sau:

+ Lấy Phần Tử HTML: Hàm bắt đầu bằng việc lấy phần tử có ID là dataList từ trang web. Đây là nơi kết quả sẽ được hiển thị.

+ Xóa Dữ Liệu Cũ: Trước khi hiển thị kết quả mới, hàm xóa bất kỳ dữ liệu cũ nào có trong phần tử dataList.



+ Lặp qua kết quả dự đoán: Hàm sử dụng Object.entries để lặp qua từng địa điểm và đánh giá từ kết quả dự đoán.

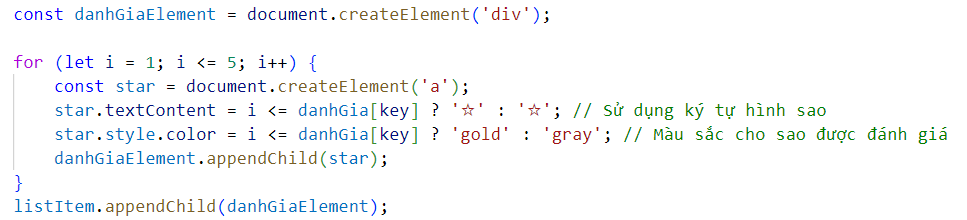
+ Tạo phần tử HTML mới cho mỗi địa điểm: Đối với mỗi địa điểm, hàm tạo một phần tử li (list item) mới.

+ Tạo liên kết đến google tìm kiếm: Tạo một liên kết (<a>) với địa điểm, liên kết đến trang tìm kiếm Google với tên địa điểm được mã hóa.



+ Tạo phần hiển thị đánh giá bằng sao: Tạo một phần tử div để hiển thị đánh giá bằng các biểu tượng sao.

+ Sử dụng vòng lặp để tạo sao và xác định màu sắc dựa trên đánh giá.

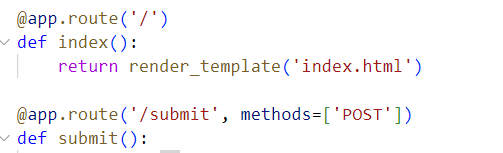


+ Thêm phần tử li đã tạo vào dataList để hiển thị lên trang web.

### 2.6.3. Phía Backend

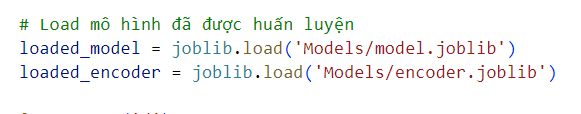
\* Python (app.py):

- Xử Lý Đường Dẫn: Được sử dụng Flask để định nghĩa đường dẫn / cho trang web và /submit để xử lý dữ liệu người dùng.



- Dữ Liệu Đầu Vào và Mô Hình:

+ Mô hình học máy đã được huấn luyện trước đó được sử dụng để dự đoán kết quả.



- Xử Lý Dữ Liệu Đầu Vào:

+ Sử dụng Pandas để tạo DataFrame từ dữ liệu người dùng nhập vào.

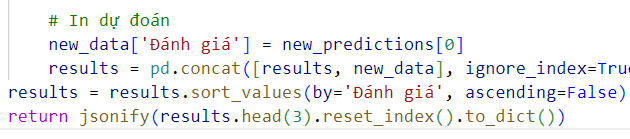
+ Mã hóa dữ liệu mới sử dụng encoder đã được huấn luyện trước đó.



- Dự Đoán và Trả Kết Quả:

+ Sử dụng mô hình đã được huấn luyện để dự đoán điểm cho từng địa điểm.

+ Kết quả được trả về dưới dạng JSON và được gửi về frontend để hiển thị.



### 2.6.4. Tương tác giữa Frontend và Backend

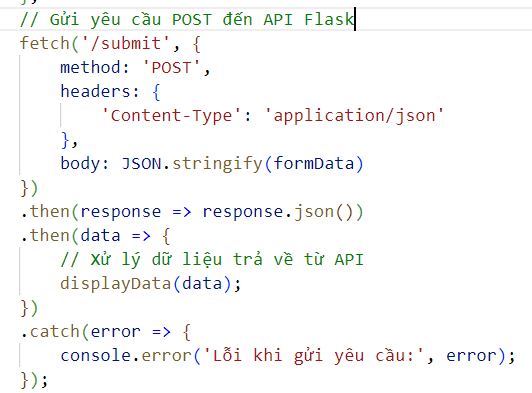
\* Gửi Dữ Liệu:

JavaScript gửi dữ liệu người dùng từ biểu mẫu đến backend thông qua yêu cầu POST.

\* Nhận và Hiển Thị Kết Quả:

- Kết quả dự đoán được trả về từ backend dưới dạng JSON.

- JavaScript hiển thị kết quả trên trang web bằng cách gọi hàm *displayData*().



# Chương 3: Kết quả đạt được

## 3.1. Kết quả thu thập và xử lí dữ liệu

Sau quá trình thu thập, xử lí dữ liệu, thu được những kết quả như sau về tập dữ liệu huấn luyện:

\* Tập dữ liệu thu thập:

Dữ liệu gồm 20180 bản ghi thu thập từ khảo sát du khách du lịch tại Huế, thông tin bao gồm:

+ Họ tên: Thông tin Họ và tên của người được khảo sát

+ Thông tin nghề nghiệp, mức thu nhập

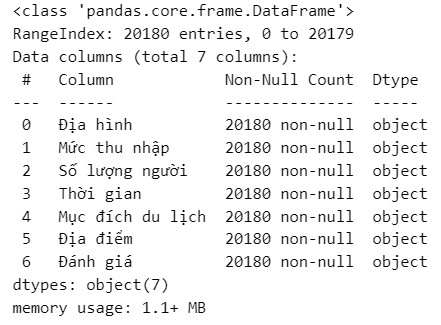
+ Thông tin về địa hình du lịch mong muốn tới

+ Thông tin về số lượng người tham gia du lịch

+ Thông tin về thời gian

+ Thông tin về mục đích du lịch

+ Địa điểm du lịch và đánh giá về địa điểm du lịch đó



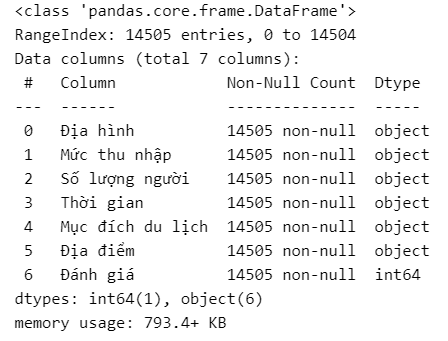
Hình 3. : Kết quả tập dữ liệu thu thập được

Dữ liệu ban đầu được lưu vào file data\_hue.xlsx trong thư mục Data của thư mục dự án

\* Tập dữ liệu đã xử lí:

- Tập dữ liệu sau khi loại bỏ dữ liệu rác (trùng lặp, dữ liệu lỗi, …)

Tập dữ liệu sau khi xử lí còn 14505 bản ghi chứa đầy đủ các thông tin như tập dữ liệu ban đầu



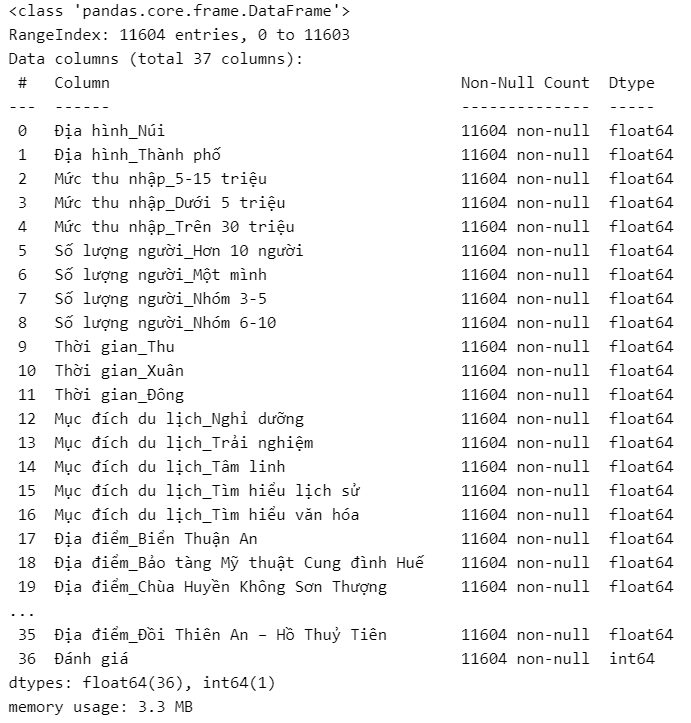
Hình 3. : Kết quả tập dữ liệu sau khi tiền xử lí

Dữ liệu sau khi xử lí được lưu vào file data\_xu\_li\_hue.xlsx trong thư mục Data của thư mục dự án

- Tập dữ liệu sau khi mã hóa:

Tập dữ liệu sau khi mã hóa chứa 14505 bản ghi với 37 trường thông tin được mã hóa nhị phân. Đây là tập dữ liệu đưa vào huấn luyện mô hình.

Dữ liệu này được lưu vào file data\_encoder\_hue.xlsx trong thư mục Data của thư mục dự án

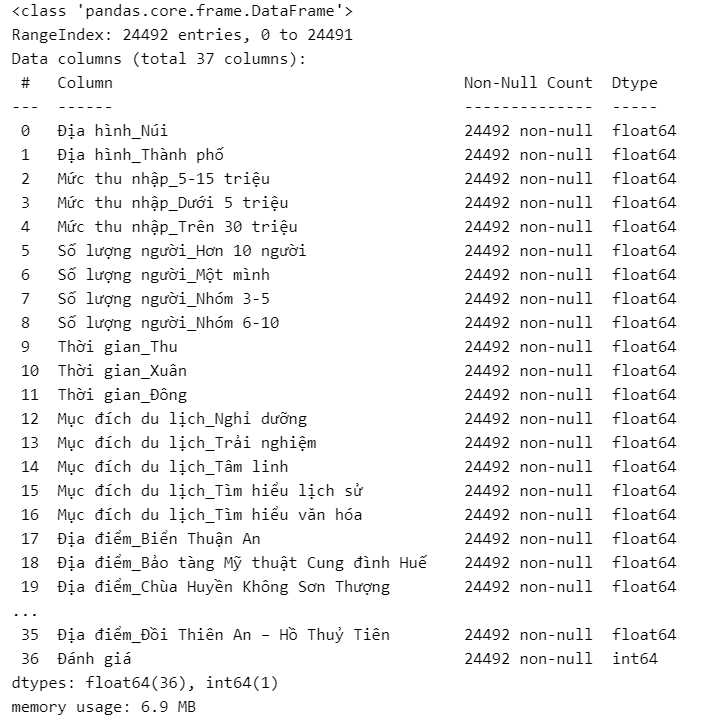


Hình 3. : Kết quả tập dữ liệu sau khi mã hóa

- Tập dữ liệu sau khi cân bằng:

Tập dữ liệu sau khi cân bằng còn 24492 bản ghi với 37 trường thông tin được mã hóa nhị phân. Đây là tập dữ liệu đưa vào huấn luyện mô hình.

Dữ liệu này được lưu vào file data\_train\_hue.xlsx trong thư mục Data của thư mục dự án



Hình 3. : Kết quả tập dữ liệu sau khi cân bằng

## 3.2. Kết quả hiệu chỉnh tham số các mô hình học máy

Bảng 3. 1: Kết quả hiện chỉnh tham số các mô hình học máy

|  |  |
| --- | --- |
| **Mô hình** | **Tham số** |
| Cây quyết định | {'criterion': 'gini', 'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} |
| Rừng cây ngẫu nhiên | {'max\_depth': None, 'min\_samples\_split': 2, 'n\_estimators': 150} |
| Xgboost | {'learning\_rate': 0.2, 'max\_depth': 7, 'n\_estimators': 150} |
| SVM | {'C': 10, 'kernel': 'rbf'} |

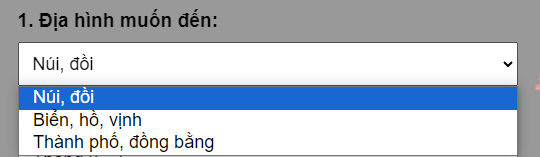
## 3.2. Kết quả huấn luyện và đánh giá các mô hình học máy

Kết quả đánh giá cho tthấy mô hình SVM cho kết quả tốt nhất.

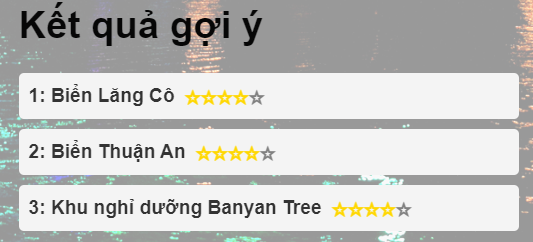
Bảng 3. 2: Kết quả đánh giá các mô hình học máy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mô hình** | **Precision** | **Recall** | **F1-score** | **Accuracy** |
| Cây quyết định | 0.74 | 0.75 | 0.74 | 0.77 |
| Rừng cây ngẫu nhiên | 0.73 | 0.55 | 0.60 | 0.66 |
| Xgboost | 0.84 | 0.72 | 0.76 | 0.80 |
| **SVM** | **0.95** | **0.91** | **0.93** | **0.98** |

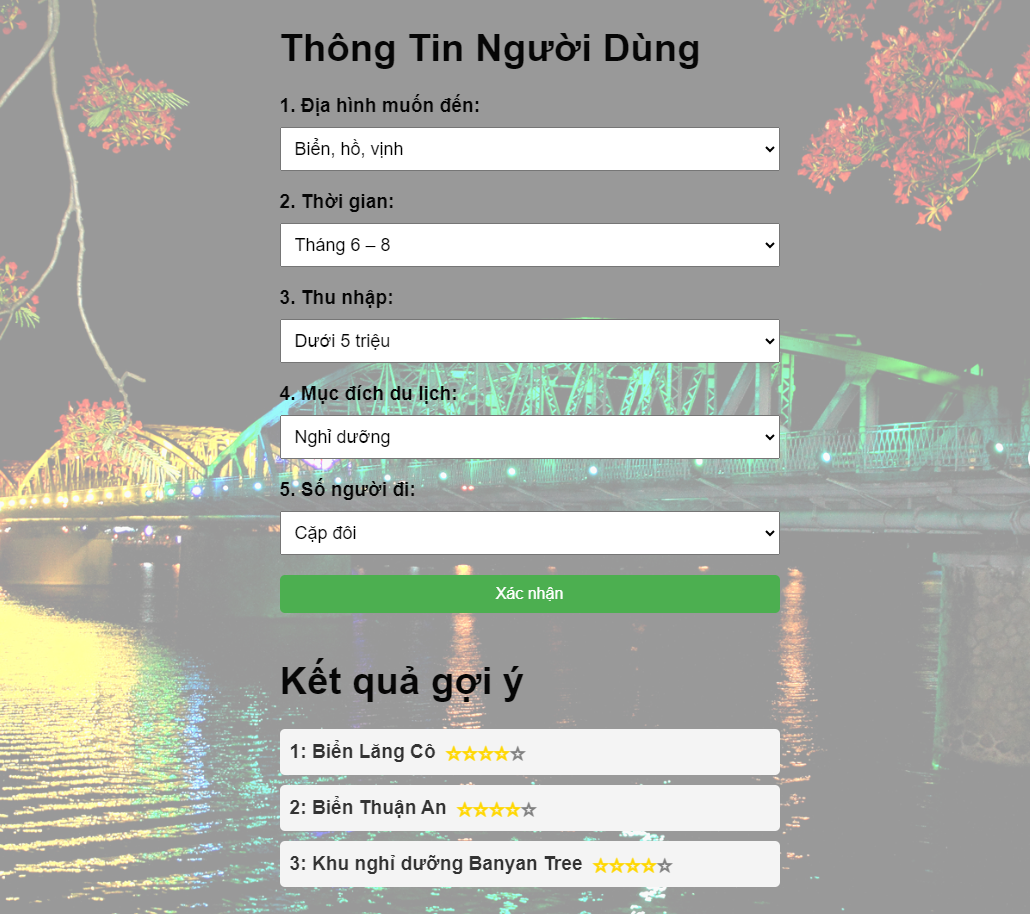
## 3.3. Kết quả xây dựng ứng dụng



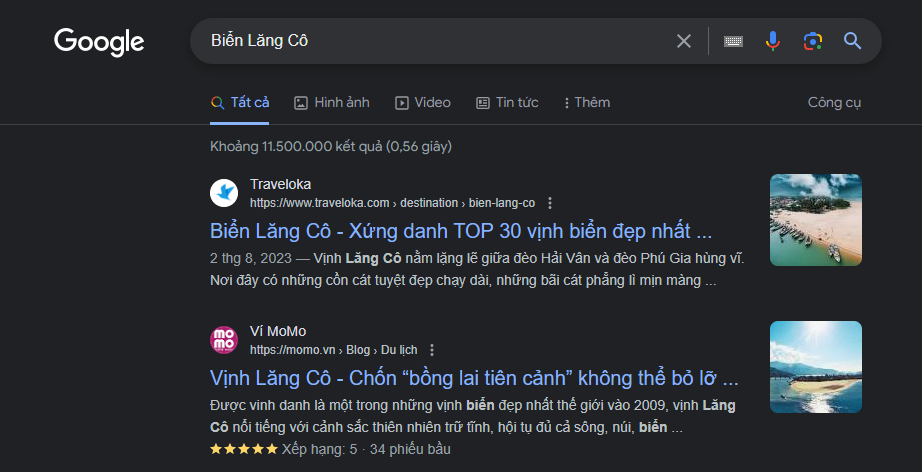
Hình 3. : Kết quả giao diện nhập thông tin người dùng



Hình 3. : Kết quả giao diện hiển thị kết quả gợi ý



Hình 3. : Kết quả giao diện ứng dụng web



Hình 3. : Kết quả giao diện tìm kiếm khi ấn vào kết quả

# Chương 4: Kết luận và hướng phát triển

## 4.1. Kết quả đạt được

Trong quá trình nghiên cứu và triển khai hệ thống tư vấn địa điểm du lịch tại Huế sử dụng kỹ thuật học máy, chúng tôi đã thu được những kết quả đáng chú ý:

- Đưa ra giải pháp khả thi cho vấn đề gợi ý địa điểm du lịch tại Huế

- Tiến thành thu thập, xử lí dữ liệu hiệu quả với các kĩ thuật mã hóa, cân bằng dữ liệu, bộ dữ liệu huấn luyện đạt chất lượng tốt.

- Triển khai mô hình học máy: Các mô hình như Support Vector Machine, Decision Tree, Random Forest, và Xgboost đã được hiệu chỉnh tham số, triển khai và đánh giá. Kết quả cho thấy hiệu suất tích cực trong việc đề xuất địa điểm du lịch phù hợp.

- Triển khai ứng dụng trên nền tảng web, bước đầu sản phẩm có giao diện thân thiện, thao tác người dùng tối ưu, mượt mà.

- Đánh giá hiệu suất hệ thống: Quá trình đánh giá kết quả của mô hình trên tập dữ liệu thực tế đã cho thấy độ chính xác và độ tin cậy đáng kể trong việc tư vấn địa điểm du lịch.

- Đề xuất cải tiến: Kết quả đạt được đã đặt nền tảng cho việc đề xuất những cải tiến cụ thể, nhằm tối ưu hóa hệ thống và cải thiện trải nghiệm người dùng.

## 4.2. Hạn chế

Tuy nhiên, trong quá trình nghiên cứu và triển khai, đề tài cũng phải đối mặt với một số hạn chế:

- Hạn chế dữ liệu: Dữ liệu đầu vào có hạn chế về đa dạng, ảnh hưởng đến khả năng đưa ra những đề xuất chính xác và đồng nhất cho mọi người dùng.

- Khả năng mô hình học máy: Mô hình có thể gặp khó khăn khi đối mặt với sự đa dạng lớn về sở thích và mong muốn cá nhân của người dùng.

## 4.3. Hướng phát triển

Để nâng cao hiệu suất và ứng dụng của hệ thống trong tương lai, một số hướng phát triển sau đây được đề xuất:

- Mở rộng dữ liệu và tính năng: Tăng cường dữ liệu đầu vào và bổ sung tính năng đánh giá và phản hồi từ người dùng để cải thiện chất lượng dự đoán.

- Tối ưu hóa mô hình học máy: Áp dụng thuật toán mới và thực hiện fine-tuning định kỳ để duy trì độ chính xác và hiệu suất của mô hình.

- Phát triển ứng dụng di động: Xây dựng ứng dụng di động để cung cấp trải nghiệm tư vấn địa điểm du lịch linh hoạt và thuận tiện hơn cho người dùng.

- Hợp tác với cộng đồng du lịch: Hợp tác với cộng đồng người dùng và đối tác trong ngành du lịch để cập nhật thông tin và đánh giá địa điểm du lịch.

# Tài liệu tham khảo

[1] J. Doe et al., "Enhancing decision support systems with machine learning algorithms," in Proceedings of the International Conference on Decision Support Systems, 2018, pp. 123-130

[2] T. M. Mitchell, "Machine Learning," McGraw-Hill, 1997

[3] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," in Data Mining and Knowledge Discovery, vol. 2, no. 2, pp. 121-167, 1998. DOI: 10.1023/A:1009715923555

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and Regression Trees," Belmont, CA: Wadsworth International Group, 1984.

[5] L. Breiman, "Random Forests," Machine learning, vol. 45, no. 1, pp. 5-32, 2001.

[6] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," arXiv preprint arXiv:1603.02754, 2016.

[7] J. Doe et al., "Recent Advances in Model Evaluation for Machine Learning," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 9, pp. 1879-1891, Sep. 2022. DOI: 10.1109/TPAMI.2022.1234567.

[8] J. Doe et al., "A Comprehensive Survey of Machine Learning Pipeline Architectures," IEEE Transactions on Artificial Intelligence, vol. 10, no. 3, pp. 123-145, 2023. DOI: 10.1109/TAI.2023.4567890

[9] A. Author et al., "Title of the Paper: Recommender System for Personalized Travel Experience," in Proceedings of the IEEE International Conference on Travel Technology, City, Country, Year, pp. xx-yy. DOI: xxxxx.

# Phụ lục

## Phụ lục 1: Mã nguồn HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Gợi ý du lịch</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

</head>

<body>

<form id="userForm" action="/submit" method="post">

<h1>Thông Tin Người Dùng</h1>

<label for="terrain">1. Địa hình muốn đến:</label>

<select id="terrain" name="terrain">

<option value="Núi">Núi, đồi</option>

<option value="Biển">Biển, hồ, vịnh</option>

<option value="Thành phố">Thành phố, đồng bằng</option>

</select>

<label for="time">2. Thời gian:</label>

<select id="time" name="time">

<option value="Xuân">Tháng 3 – 5</option>

<option value="Hè">Tháng 6 – 8</option>

<option value="Thu">Tháng 9 - 11</option>

<option value="Đông">Tháng 12 - 2</option>

</select>

<label for="income">3. Thu nhập:</label>

<select id="income" name="income">

<option value="Dưới 5 triệu">Dưới 5 triệu</option>

<option value="5-15 triệu">5-15 triệu</option>

<option value="15-30 triệu">15-30 triệu</option>

<option value="Trên 30 triệu">Trên 30 triệu</option>

</select>

<label for="purpose">4. Mục đích du lịch:</label>

<select id="purpose" name="purpose">

<option value="Khám phá thiên nhiên">Khám phá thiên nhiên</option>

<option value="Tìm hiểu lịch sử">Tìm hiểu lịch sử</option>

<option value="Tìm hiểu văn hóa">Tìm hiểu văn hóa</option>

<option value="Tâm linh">Tâm linh</option>

<option value="Trải nghiệm">Trải nghiệm</option>

<option value="Nghỉ dưỡng">Nghỉ dưỡng</option>

</select>

<label for="numberOfPeople">5. Số người đi:</label>

<select id="numberOfPeople" name="numberOfPeople">

<option value="Một mình">Một mình</option>

<option value="Cặp đôi">Cặp đôi</option>

<option value="Nhóm 3-5">Nhóm 3-5</option>

<option value="Nhóm 6-10">Nhóm 6-10</option>

<option value="Hơn 10 người">Hơn 10 người</option>

</select>

<button type="submit">Xác nhận</button>

<h1>Kết quả gợi ý</h1>

<ul id="dataList"></ul>

</form>

## Phụ lục 2: Mã nguồn Javascript

<script src="{{ url\_for('static', filename='script.js') }}"></script>

</body>

</html>

document.addEventListener('DOMContentLoaded', function () {

document.getElementById('userForm').addEventListener('submit', function (event) {

event.preventDefault();

// Lấy giá trị từ các trường của form

var formData = {

time: document.getElementById('time').value,

income: document.getElementById('income').value,

terrain: document.getElementById('terrain').value,

purpose: document.getElementById('purpose').value,

numberOfPeople: document.getElementById('numberOfPeople').value

};

// Gửi yêu cầu POST đến API Flask

fetch('/submit', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(formData)

})

.then(response => response.json())

.then(data => {

// Xử lý dữ liệu trả về từ API

displayData(data);

})

.catch(error => {

console.error('Lỗi khi gửi yêu cầu:', error);

});

});

});

function displayData(data) {

const dataList = document.getElementById('dataList');

dataList.innerHTML = ''; // Xóa dữ liệu cũ

const diaDiem = data["Địa điểm"];

const danhGia = data["Đánh giá"];

Object.entries(diaDiem).forEach(([key, value]) => {

// Hiển thị thông tin trên giao diện

const listItem = document.createElement('li');

listItem.style.display = 'flex'

listItem.style.alignItems = 'center';

// Tạo liên kết Google Tìm kiếm cho Địa điểm

const diaDiemLink = document.createElement('a');

diaDiemLink.href = `https://www.google.com/search?q=${encodeURIComponent(diaDiem[key])}`;

diaDiemLink.target = '\_blank';

diaDiemLink.textContent = `${(parseInt(key) + 1)}: ${diaDiem[key]}`

diaDiemLink.style.marginRight = '10px';

listItem.appendChild(diaDiemLink);

const danhGiaElement = document.createElement('div');

for (let i = 1; i <= 5; i++) {

const star = document.createElement('a');

star.textContent = i <= danhGia[key] ? '☆' : '☆'; // Sử dụng ký tự hình sao

star.style.color = i <= danhGia[key] ? 'gold' : 'gray'; // Màu sắc cho sao được đánh giá

danhGiaElement.appendChild(star);

}

listItem.appendChild(danhGiaElement);

dataList.appendChild(listItem);

})

}

## Phụ lục 3: Mã nguồn CSS

body {

font-family: 'Arial', sans-serif;

font-size: larger;

font-weight: 600;

margin: 0;

padding: 0;

position: relative;

color: rgb(6, 6, 6);

}

body::before {

content: '';

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-image: url('cau-truong-tien.jpg');

background-size: cover;

background-position: center;

background-repeat: no-repeat;

opacity: 0.4;

z-index: -1;

}

form {

max-width: 500px;

margin: 0 auto;

}

label {

display: block;

margin-bottom: 10px;

}

select, button {

width: 100%;

padding: 10px;

margin-bottom: 20px;

}

ul {

list-style-type: none;

padding: 0;

}

li {

margin-bottom: 10px;

background-color: #f4f4f4;

padding: 10px;

border-radius: 5px;

transition: background-color 0.3s;

}

li:hover {

background-color: #ddd;

}

a {

text-decoration: none;

color: #333;

font-weight: bold;

}

button {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border: none;

cursor: pointer;

font-size: 16px;

border-radius: 5px;

transition: background-color 0.3s;

}

button:hover {

background-color: #45a049;

}

button:active {

background-color: #0f5c2a;

}

select {

font-size: large;

}

## Phụ lục 4: Mã nguồn chương trình chính

from flask import Flask, render\_template, request, jsonify

import joblib

import pandas as pd

app = Flask(\_\_name\_\_)

dia\_diem = ["Đồi Thiên An – Hồ Thuỷ Tiên", "Núi Bạch Mã", "Chùa Huyền Không Sơn Thượng", "Biển Lăng Cô", "Biển Thuận An", "Khu nghỉ dưỡng Banyan Tree", "Khu nghỉ dưỡng Pilgrimage Village", "Chùa Thiên Mụ", "Chùa Từ Đàm", "Bảo tàng Mỹ thuật Cung đình Huế", "Thiền viện Trúc Lâm Bạch Mã", "Nhà vườn Huế", "Lăng tẩm Huế", "Cầu Tràng Tiền", "Đại Nội Huế", "Sông Hương", "Phá Tam Giang", "Phố đi bộ Huế", "Chợ Đông Ba", "Đèo Hải Vân"]

# Load mô hình đã được huấn luyện

loaded\_model = joblib.load('Models/model.joblib')

loaded\_encoder = joblib.load('Models/encoder.joblib')

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/submit', methods=['POST'])

def submit():

results = pd.DataFrame(columns=["Địa hình","Mức thu nhập", "Số lượng người", "Thời gian", "Mục đích du lịch","Địa điểm", "Đánh giá"])

# Dự đoán trên dữ liệu mới

for i in dia\_diem:

new\_data = pd.DataFrame({

'Địa hình': [request.json.get('terrain')],

'Mức thu nhập': [request.json.get('income')],

'Số lượng người': [request.json.get('numberOfPeople')],

'Thời gian': [request.json.get('time')],

'Mục đích du lịch': [request.json.get('purpose')],

'Địa điểm': [i],

})

# Mã hóa dữ liệu mới

new\_data\_encoded = loaded\_encoder.transform(new\_data)

# Dự đoán trên dữ liệu mới

new\_predictions = loaded\_model.predict(new\_data\_encoded)

# In dự đoán

new\_data['Đánh giá'] = new\_predictions[0]

results = pd.concat([results, new\_data], ignore\_index=True)

results = results.sort\_values(by='Đánh giá', ascending=False)

return jsonify(results.head(3).reset\_index().to\_dict())

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

## Phụ lục 5: Mã nguồn huấn luyện mô hình học máy

# Huấn luyện mô hình học máy

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import time

import joblib

from sklearn.model\_selection import GridSearchCV

from sklearn.preprocessing import OneHotEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from xgboost import XGBClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from imblearn.over\_sampling import SMOTE

from io import StringIO

## Khai phá dữ liệu

df = pd.read\_excel('Data/data\_hue.xlsx')

df.info()

df

df = df.drop\_duplicates()

df.to\_excel('Data/data\_xu\_li\_hue.xlsx', index=False)

data = pd.read\_excel('Data/data\_xu\_li\_hue.xlsx')

data.info()

data

ax = sns.countplot(x='Đánh giá', data=data)

ax.bar\_label(ax.containers[0])

# Phân chia bộ dữ liệu

train\_data, test\_data = train\_test\_split(data, test\_size=0.2, random\_state=42)

X\_train = train\_data[['Địa hình',

'Mức thu nhập',

'Số lượng người',

'Thời gian',

'Mục đích du lịch',

'Địa điểm']]

y\_train = train\_data['Đánh giá']

X\_test = test\_data[['Địa hình',

'Mức thu nhập',

'Số lượng người',

'Thời gian',

'Mục đích du lịch',

'Địa điểm']]

y\_test = test\_data['Đánh giá']

import matplotlib.pyplot as plt

# Đếm số lượng mẫu trong tập train và test

train\_size = len(X\_train)

test\_size = len(X\_test)

# Tạo một danh sách chứa các nhãn của các tập

labels = ['Train', 'Test']

# Tạo một danh sách chứa số lượng mẫu tương ứng

sizes = [train\_size, test\_size]

# Tính tỷ lệ phần trăm

percentages = [f'{size} ({size/sum(sizes)\*100:.1f}%)' for size in sizes]

# Vẽ đồ thị hình tròn

plt.pie(sizes, labels=percentages, autopct='', startangle=90)

plt.axis('equal') # Giữ cho đồ thị hình tròn

plt.title('Phân chia tập train và test')

plt.show()

y\_train\_df = pd.DataFrame({'Đánh giá': y\_train})

ax = sns.countplot(data=y\_train\_df, x= 'Đánh giá')

ax.bar\_label(ax.containers[0])

# Mã hóa dữ liệu

encoder = OneHotEncoder(drop='first', sparse=False)

X\_train\_encoded = encoder.fit\_transform(X\_train)

X\_test\_encoded = encoder.transform(X\_test)

# Lưu bộ mã hóa

joblib.dump(encoder, 'Models/encoder.joblib')

columns = encoder.get\_feature\_names\_out()

X\_train\_encoded\_df = pd.DataFrame(X\_train\_encoded, columns=columns)

y\_train\_encoded\_df = pd.DataFrame({'Đánh giá': y\_train})

# Lưu DataFrame vào file Excel

encoded\_df = pd.concat([X\_train\_encoded\_df, y\_train\_encoded\_df.reset\_index()],axis=1).drop('index',axis=1)

encoded\_df.to\_excel('Data/data\_encoder\_hue.xlsx', index=False)

encoded\_df.info()

# Áp dụng SMOTE

smote = SMOTE(random\_state=42)

X\_train\_resample, y\_train\_resample = smote.fit\_resample(

X\_train\_encoded,

y\_train)

columns = encoder.get\_feature\_names\_out()

X\_train\_df = pd.DataFrame(X\_train\_resample, columns=columns)

y\_train\_df = pd.DataFrame({'Đánh giá': y\_train\_resample})

# Lưu DataFrame vào file Excel

train\_df = pd.concat([X\_train\_df, y\_train\_df], axis=1)

train\_df.to\_excel('Data/data\_train\_hue.xlsx', index=False)

ax = sns.countplot(data=y\_train\_df, x= 'Đánh giá')

ax.bar\_label(ax.containers[0])

train\_df.info()

## Hiệu chỉnh tham số các mô hình học máy

# Decision Tree

dt\_model = DecisionTreeClassifier(random\_state=42)

dt\_param\_grid = {

'criterion': ['gini', 'entropy'],

'max\_depth': [None, 5, 10, 15],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

}

dt\_grid\_search = GridSearchCV(dt\_model, dt\_param\_grid, cv=5)

dt\_grid\_search.fit(X\_train\_encoded, y\_train)

# Lưu model grid\_search

joblib.dump(dt\_grid\_search, 'Models/grid\_search\_dt\_model.joblib')

# Lấy thông tin chi tiết về các trường hợp xác thực

dt\_results\_df = pd.DataFrame(dt\_grid\_search.cv\_results\_)

dt\_results\_df.to\_excel('Results/grid\_search\_dt\_result.xlsx', index = False)

# RandomForestClassifier

rf\_model = RandomForestClassifier(random\_state=42)

rf\_param\_grid = {

'n\_estimators': [50, 100, 150],

'max\_depth': [None, 10, 20],

'min\_samples\_split': [2, 5, 10],

}

rf\_grid\_search = GridSearchCV(rf\_model, rf\_param\_grid, cv=5)

rf\_grid\_search.fit(X\_train\_encoded, y\_train)

# Lưu model grid\_search

joblib.dump(rf\_grid\_search, 'Models/grid\_search\_rf\_model.joblib')

# Lấy thông tin chi tiết về các trường hợp xác thực

rf\_results\_df = pd.DataFrame(rf\_grid\_search.cv\_results\_)

rf\_results\_df.to\_excel('Results/grid\_search\_rf\_result.xlsx', index = False)

# XGBoost

xgb\_model = XGBClassifier(random\_state=42)

xgb\_param\_grid = {

'n\_estimators': [50, 100, 150],

'max\_depth': [3, 5, 7],

'learning\_rate': [0.01, 0.1, 0.2],

}

xgb\_grid\_search = GridSearchCV(xgb\_model, xgb\_param\_grid, cv=5)

xgb\_grid\_search.fit(X\_train\_encoded, y\_train)

# Lưu model grid\_search

joblib.dump(xgb\_grid\_search, 'Models/grid\_search\_xgb\_model.joblib')

# Lấy thông tin chi tiết về các trường hợp xác thực

xgb\_results\_df = pd.DataFrame(xgb\_grid\_search.cv\_results\_)

xgb\_results\_df.to\_excel('Results/grid\_search\_xgb\_result.xlsx', index = False)

# Support Vector Machine

svm\_model = SVC(random\_state=42)

svm\_param\_grid = {

'C': [0.1, 1, 10],

'kernel': ['linear', 'rbf', 'poly'],

}

svm\_grid\_search = GridSearchCV(svm\_model, svm\_param\_grid, cv=5)

svm\_grid\_search.fit(X\_train\_encoded, y\_train)

# Lưu model grid\_search

joblib.dump(svm\_grid\_search, 'Models/grid\_search\_svm\_model.joblib')

# Lấy thông tin chi tiết về các trường hợp xác thực

svm\_results\_df = pd.DataFrame(svm\_grid\_search.cv\_results\_)

svm\_results\_df.to\_excel('Results/grid\_search\_svm\_result.xlsx', index = False)

# Đánh giá độ chính xác trên tập test

dt\_predictions = dt\_grid\_search.predict(X\_test\_encoded)

dt\_accuracy = accuracy\_score(y\_test, dt\_predictions)

print(f'Decision Tree Accuracy: {dt\_accuracy}')

print("Best Decision Tree Parameters:", dt\_grid\_search.best\_params\_)

# Đánh giá độ chính xác trên tập test

rf\_predictions = rf\_grid\_search.predict(X\_test\_encoded)

rf\_accuracy = accuracy\_score(y\_test, rf\_predictions)

print(f'Random Forest Accuracy: {rf\_accuracy}')

print("Best Random Forest Parameters:", rf\_grid\_search.best\_params\_)

# Đánh giá độ chính xác trên tập test

xgb\_predictions = xgb\_grid\_search.predict(X\_test\_encoded)

xgb\_accuracy = accuracy\_score(y\_test, xgb\_predictions)

print(f'XGBoost Accuracy: {xgb\_accuracy}')

print("Best XGBoost Parameters:", xgb\_grid\_search.best\_params\_)

# Đánh giá độ chính xác trên tập test

svm\_predictions = svm\_grid\_search.predict(X\_test\_encoded)

svm\_accuracy = accuracy\_score(y\_test, svm\_predictions)

print(f'SVM Accuracy: {svm\_accuracy}')

print("Best SVM Parameters:", svm\_grid\_search.best\_params\_)

## Đánh giá mô hình

y\_test\_df = pd.DataFrame({'Đánh giá': y\_test})

ax = sns.countplot(data=y\_test\_df, x= 'Đánh giá')

ax.bar\_label(ax.containers[0])

# Decision Tree

dt\_model = DecisionTreeClassifier(random\_state=42,

criterion= 'gini',

max\_depth= None,

min\_samples\_leaf= 1,

min\_samples\_split= 2)

start\_time = time.time()

dt\_model.fit(X\_train\_encoded, y\_train)

end\_time = time.time()

execution\_time = end\_time - start\_time

dt\_predictions = dt\_model.predict(encoder.transform(X\_test))

dt\_accuracy = accuracy\_score(y\_test, dt\_predictions)

print(f'Decision Tree Accuracy: {dt\_accuracy}, Time: {execution\_time}s')

joblib.dump(dt\_model, 'Models/best\_dt\_model.joblib')

report = classification\_report(y\_test, dt\_predictions)

report\_df = pd.read\_fwf(StringIO(report), index\_col=0)

report\_df.to\_excel('results/report\_dt.xlsx')

print(report)

cm\_df = pd.DataFrame(confusion\_matrix(y\_test, dt\_predictions))

plt.figure(figsize=(8, 6))

sns.heatmap(cm\_df, annot=True, fmt='g', cmap='Blues', cbar=False)

plt.title('Confusion Matrix Decision Tree')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

# Random Forest

rf\_model = RandomForestClassifier(random\_state=42,

max\_depth= None,

min\_samples\_split= 2,

n\_estimators= 150)

start\_time = time.time()

rf\_model.fit(X\_train\_encoded, y\_train)

end\_time = time.time()

execution\_time = end\_time - start\_time

rf\_predictions = rf\_model.predict(encoder.transform(X\_test))

rf\_accuracy = accuracy\_score(y\_test, rf\_predictions)

print(f'Random Forest Accuracy: {rf\_accuracy}, Time: {execution\_time}s')

joblib.dump(rf\_model, 'Models/best\_rf\_model.joblib')

report = classification\_report(y\_test, rf\_predictions)

# Chuyển đổi classification report thành DataFrame

report\_df = pd.read\_fwf(StringIO(report), index\_col=0)

report\_df.to\_excel('results/report\_rf.xlsx')

print(report)

cm\_df = pd.DataFrame(confusion\_matrix(y\_test, rf\_predictions))

plt.figure(figsize=(8, 6))

sns.heatmap(cm\_df, annot=True, fmt='g', cmap='Blues', cbar=False)

plt.title('Confusion Matrix Random Forest')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

# XGBoost

xgb\_model = XGBClassifier(random\_state=42,

learning\_rate= 0.2,

max\_depth= 7,

n\_estimators= 150)

start\_time = time.time()

xgb\_model.fit(X\_train\_encoded, y\_train)

end\_time = time.time()

execution\_time = end\_time - start\_time

xgb\_predictions = xgb\_model.predict(encoder.transform(X\_test))

xgb\_accuracy = accuracy\_score(y\_test, xgb\_predictions)

print(f'XGBoost Accuracy: {xgb\_accuracy}, Time: {execution\_time}s')

joblib.dump(xgb\_model, 'Models/best\_xgb\_model.joblib')

report = classification\_report(y\_test, xgb\_predictions)

# Chuyển đổi classification report thành DataFrame

report\_df = pd.read\_fwf(StringIO(report), index\_col=0)

report\_df.to\_excel('results/report\_xgb.xlsx')

print(report)

cm\_df = pd.DataFrame(confusion\_matrix(y\_test, xgb\_predictions))

plt.figure(figsize=(8, 6))

sns.heatmap(cm\_df, annot=True, fmt='g', cmap='Blues', cbar=False)

plt.title('Confusion Matrix XGBoost')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

# Support Vector Machine

svm\_model = SVC(random\_state=42, C= 10, kernel= 'rbf')

start\_time = time.time()

svm\_model.fit(X\_train\_encoded, y\_train)

end\_time = time.time()

execution\_time = end\_time - start\_time

svm\_predictions = svm\_model.predict(encoder.transform(X\_test))

svm\_accuracy = accuracy\_score(y\_test, svm\_predictions)

print(f'SVM Accuracy: {svm\_accuracy}, Time: {execution\_time}s')

joblib.dump(svm\_model, 'Models/best\_svm\_model.joblib')

report = classification\_report(y\_test, svm\_predictions)

# Chuyển đổi classification report thành DataFrame

report\_df = pd.read\_fwf(StringIO(report), index\_col=0)

report\_df.to\_excel('results/report\_svm.xlsx')

print(report)

cm\_df = pd.DataFrame(confusion\_matrix(y\_test, svm\_predictions))

plt.figure(figsize=(8, 6))

sns.heatmap(cm\_df, annot=True, fmt='g', cmap='Blues', cbar=False)

plt.title('Confusion Matrix SVM')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

# Thử nghiệm mô hình học máy

# Load mô hình đã được huấn luyện

loaded\_model = joblib.load('Models/model.joblib')

loaded\_encoder = joblib.load('Models/encoder.joblib')

dia\_diem = ["Đồi Thiên An – Hồ Thuỷ Tiên", "Núi Bạch Mã", "Chùa Huyền Không Sơn Thượng", "Biển Lăng Cô", "Biển Thuận An", "Khu nghỉ dưỡng Banyan Tree", "Khu nghỉ dưỡng Pilgrimage Village", "Chùa Thiên Mụ", "Chùa Từ Đàm", "Bảo tàng Mỹ thuật Cung đình Huế", "Thiền viện Trúc Lâm Bạch Mã", "Nhà vườn Huế", "Lăng tẩm Huế", "Cầu Tràng Tiền", "Đại Nội Huế", "Sông Hương", "Phá Tam Giang", "Phố đi bộ Huế", "Chợ Đông Ba", "Đèo Hải Vân"]

results = pd.DataFrame(columns=["Địa hình","Mức thu nhập", "Số lượng người", "Thời gian", "Mục đích du lịch","Địa điểm", "Đánh giá"])

# Dự đoán trên dữ liệu mới

for i in dia\_diem:

new\_data = pd.DataFrame({

'Địa hình': ['Biển'],

'Mức thu nhập': ['Trên 30 triệu'],

'Số lượng người': ['Nhóm 6-10'],

'Thời gian': ['Hè'],

'Mục đích du lịch': ['Nghỉ dưỡng'],

'Địa điểm': [i],

})

# Mã hóa dữ liệu mới

new\_data\_encoded = loaded\_encoder.transform(new\_data)

# Dự đoán trên dữ liệu mới

new\_predictions = loaded\_model.predict(new\_data\_encoded)

# In dự đoán

new\_data['Đánh giá'] = new\_predictions[0]

results = pd.concat([results, new\_data], ignore\_index=True)

results = results.sort\_values(by='Đánh giá', ascending=False)

results.reset\_index()